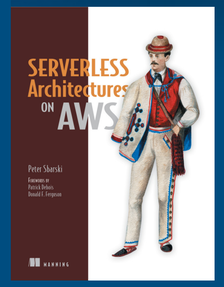
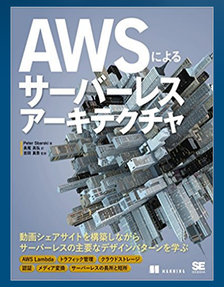
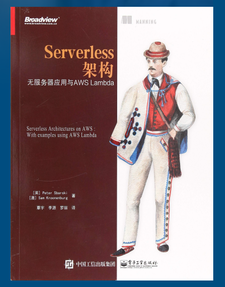
Dancing with Serverless

Peter Sbarski | @sbarski

About Me



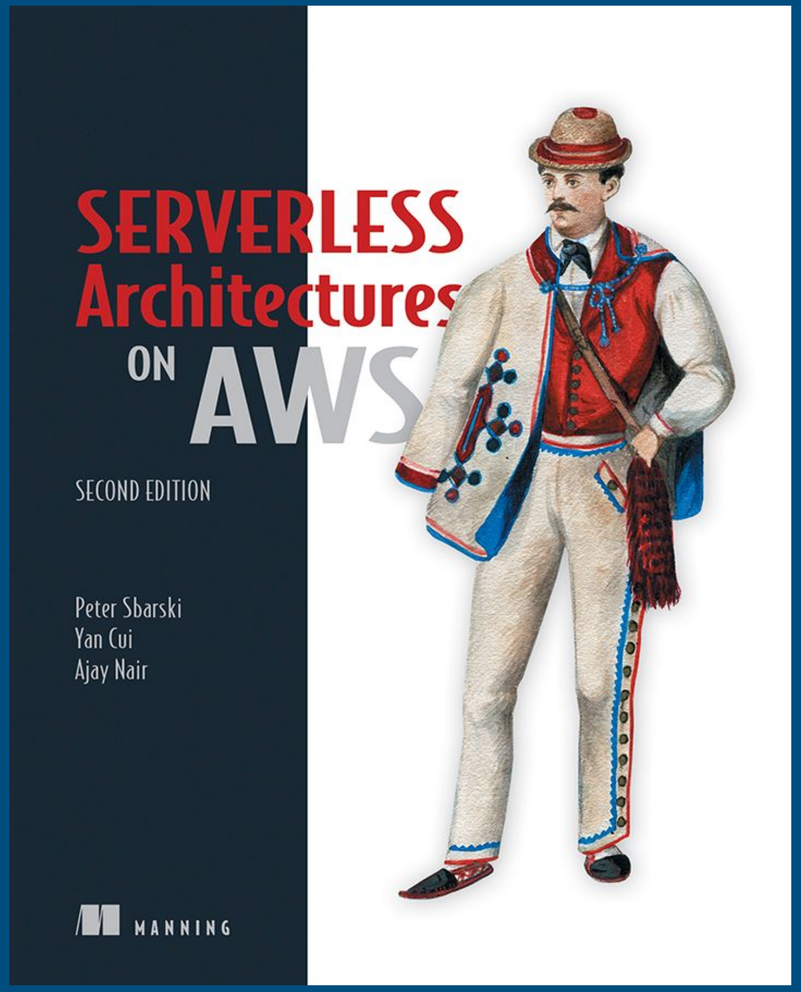


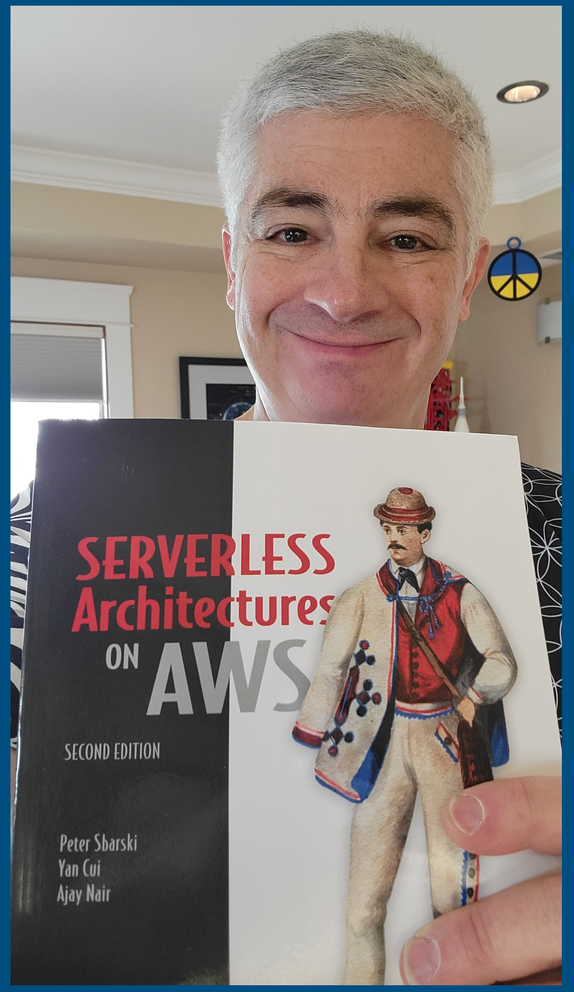


- AWS Serverless Hero
- Author of Serverless Architectures on AWS
- Organiser of the Melbourne Serverless Meetup
- Former VP Education & Research at A Cloud Guru
- Former head of Serverlessconf

- Building something new… reveal later

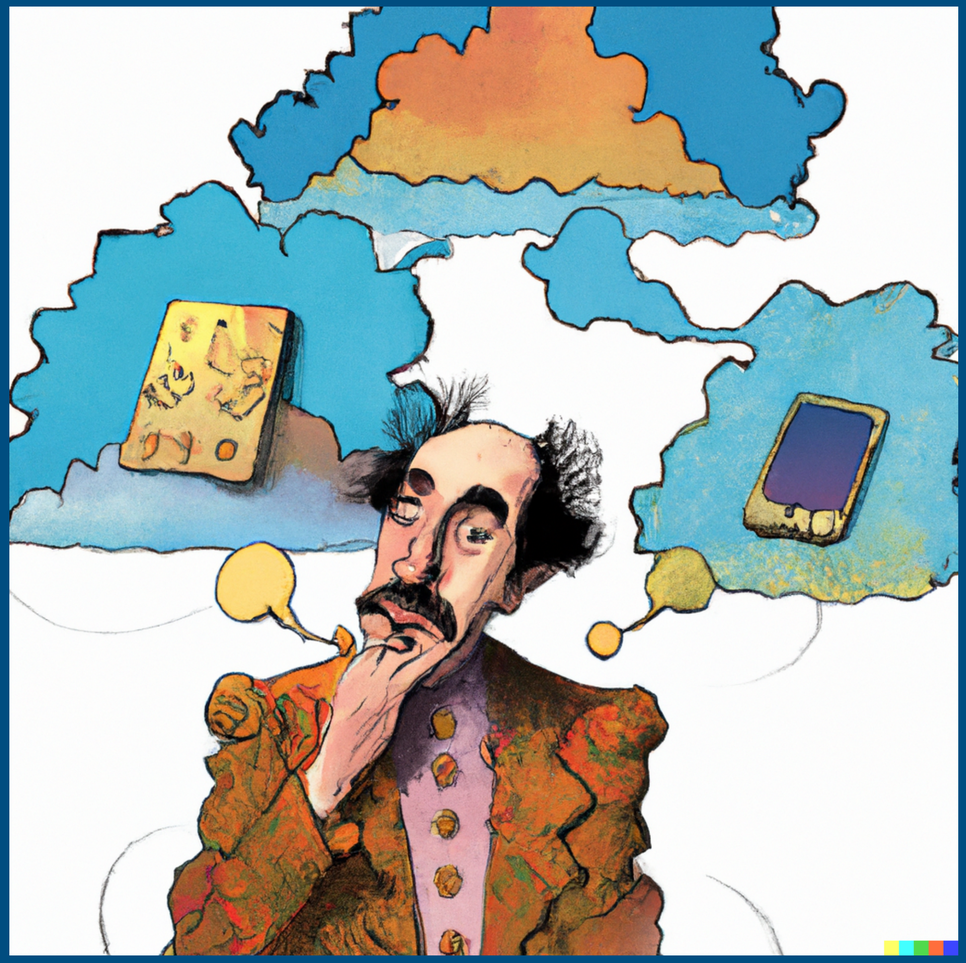
Serverless Architectures on AWS (2nd Edition)





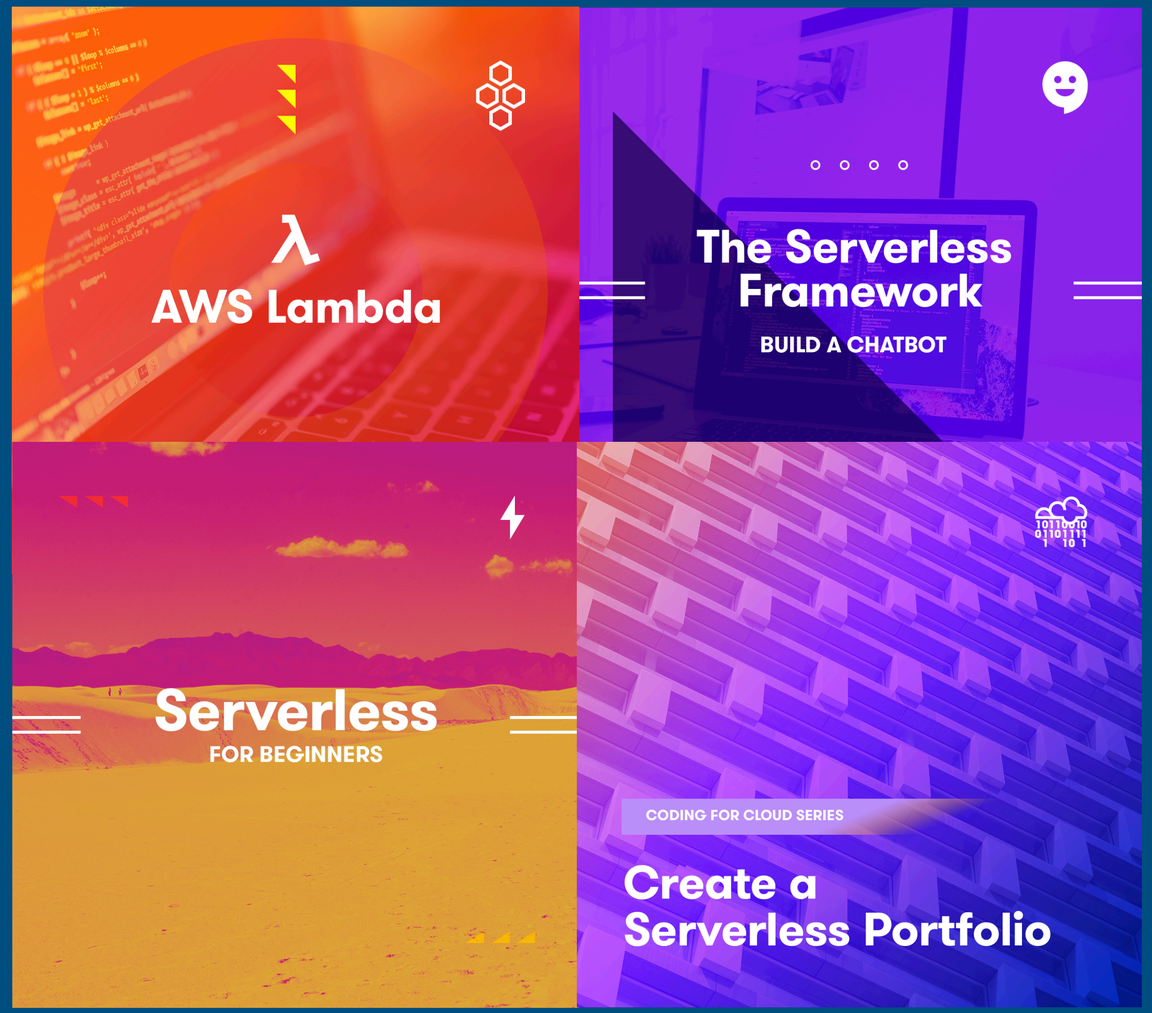
I am thinking about this talk

According to Dall E



Early Days at ACG

Teaching the world to cloud



Video Lessons

Quiz Engine

Online Store

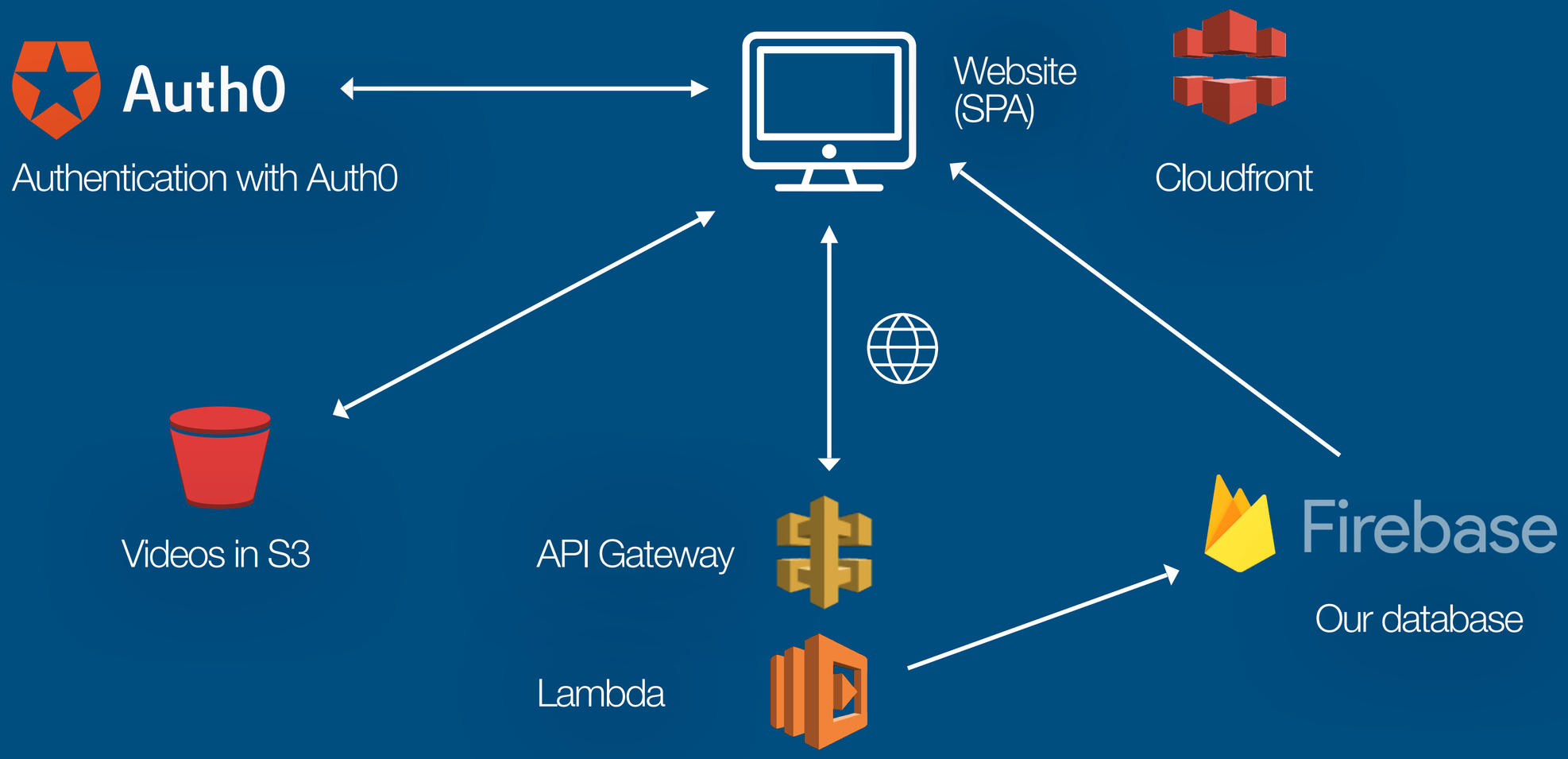
Sign Up / Login

Scale Effortlessly

Low Operational Overhead

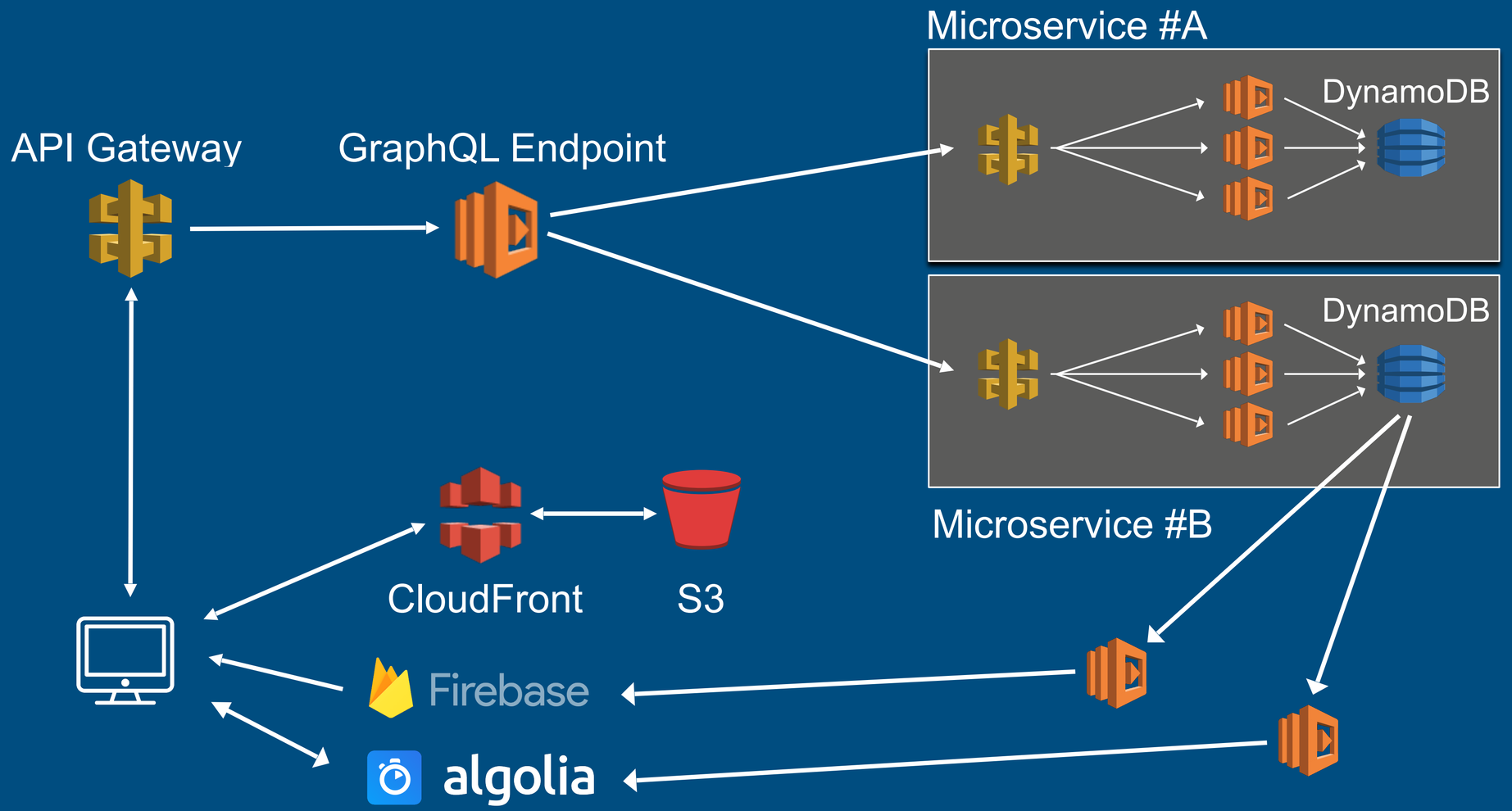
Starting up in 2015

Serverless Monolith



Evolution

Teams of developers working in parallel



We evolved our architecture

And the costs weren't too bad either

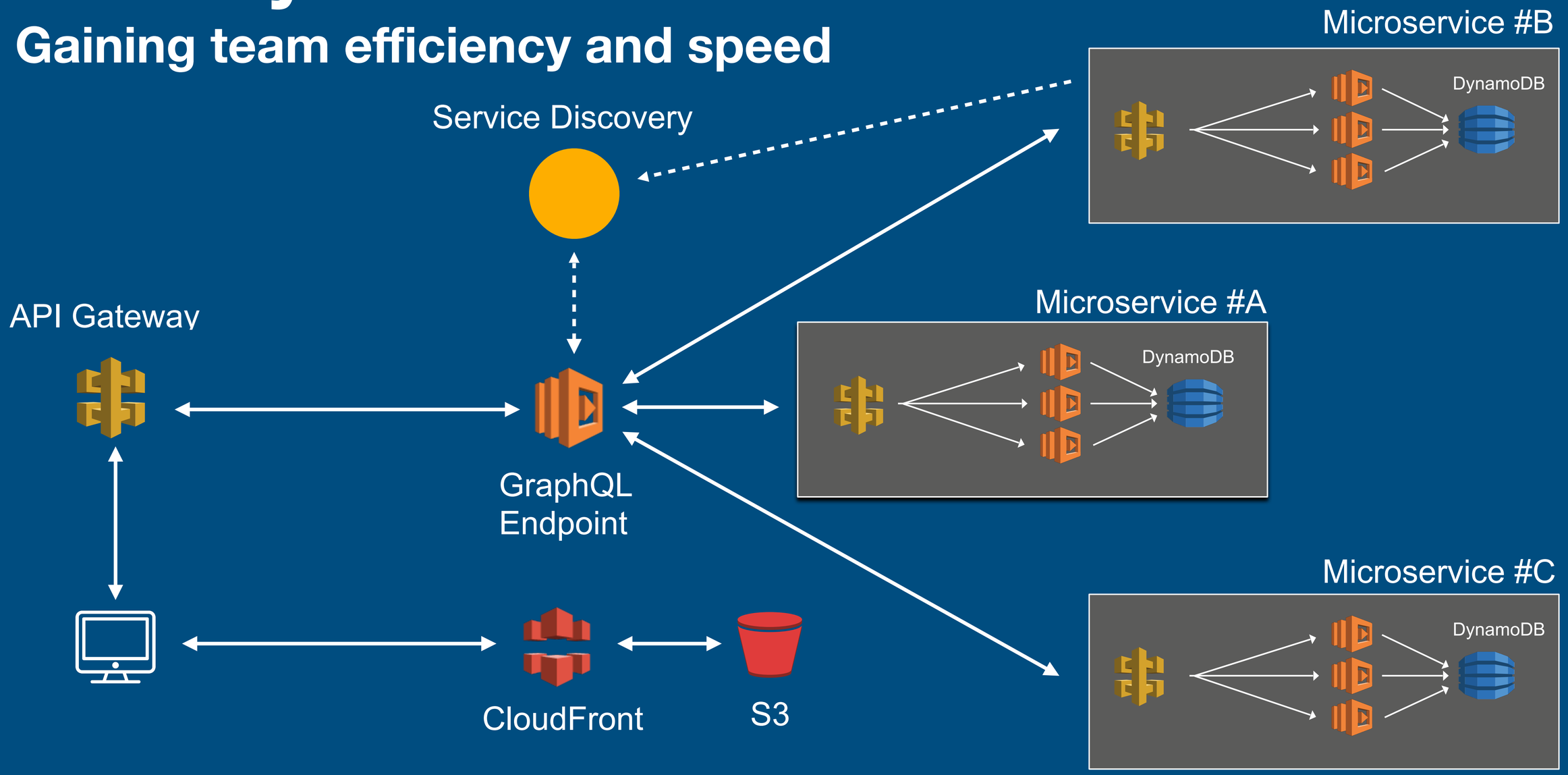
- 289 Lambda Functions
- 19 Micro-services
- 3.68TB of data in S3

- 107m Lambda Invocations / month
- 45m API Requests / month
- 3.8+ TB of data via CloudFront **per day**
- 650K+ users

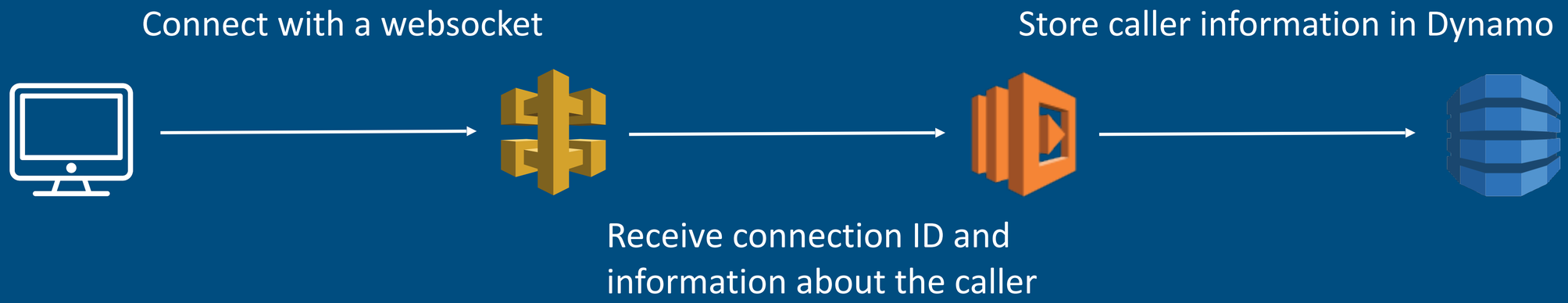
Service	Cost
Key Management Service	$25.26
Simple Storage Service (S3)	$108.23
Config	$109.84
Elastic Transcoder	$154.17
API Gateway	**$192.14**
Developer Support	$314.59
Redshift	$371.50
DynamoDB	$373.54
Lambda	**$706.49**
CloudWatch	$3,142.73
CloudFront	$5,099.85

Maturity

Gaining team efficiency and speed



Web Sockets
With API Gateway



Mullet Architecture

With thanks to Tim Wagner

Common Benefits

When things go right

- It’s fast to build (shortened time to market)
- Massive scale and initially can be cheap or free
- It’s operationally efficient
- It’s not Kubernetes
- Large architectural pivots/changes are possible
- It’s fun - developers love it

Common Complaints

Why can’t things just be easy

- Hard to dev locally
- Hard to debug
- Hard to observe and monitor
- Hard or impossible to do certain things (e.g. long-running tasks)
- Lock-in is a problem. Maybe?

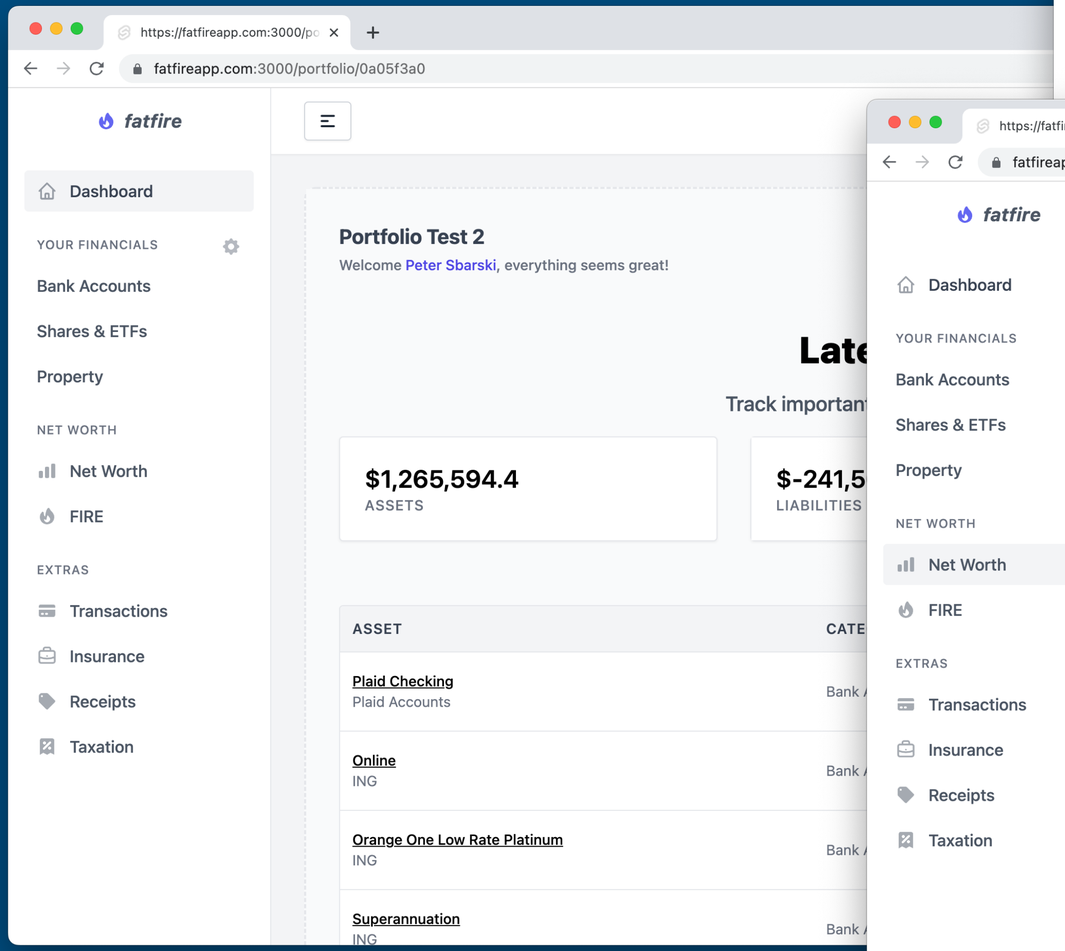
How I build today

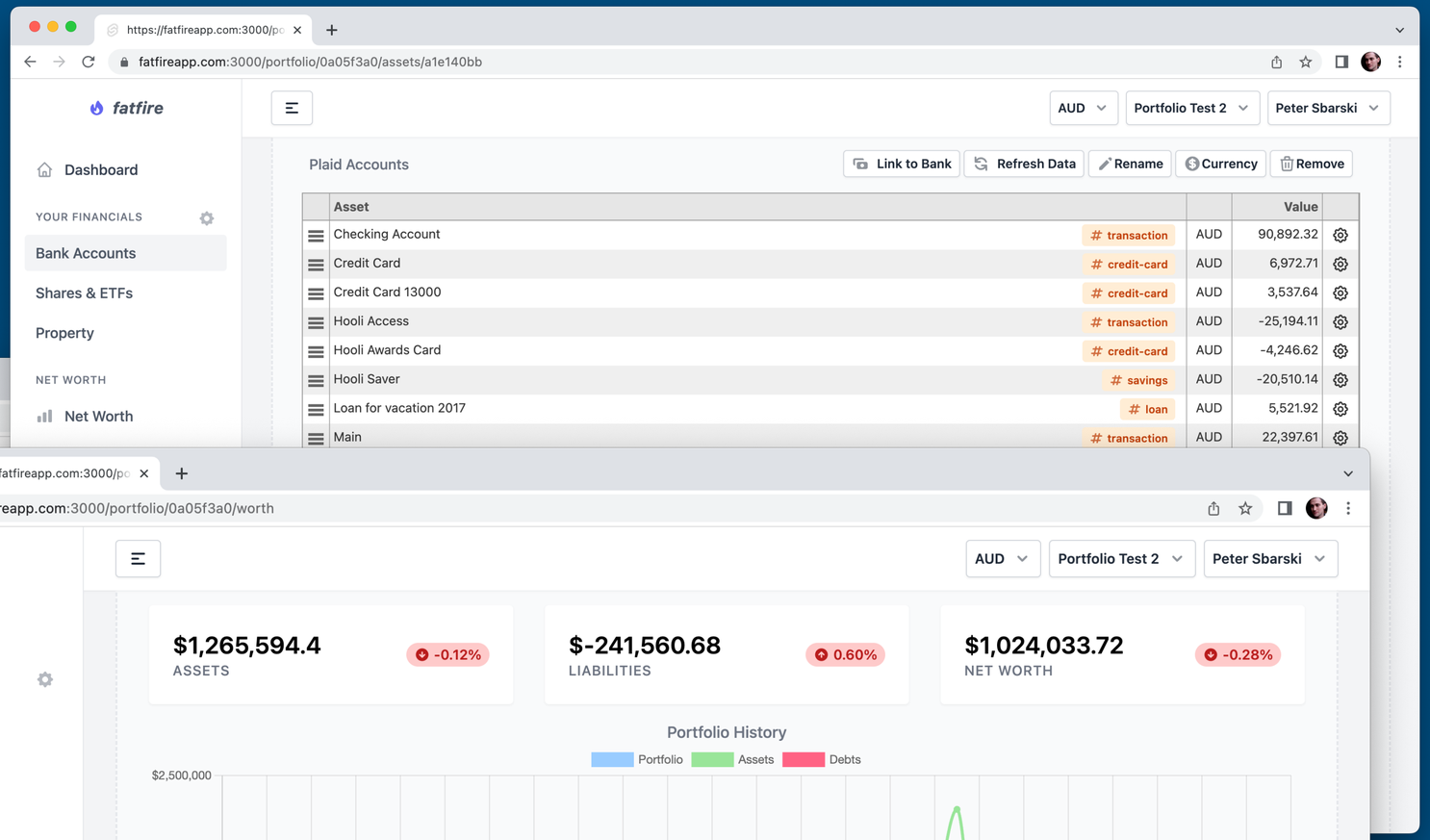
Back to working on my own

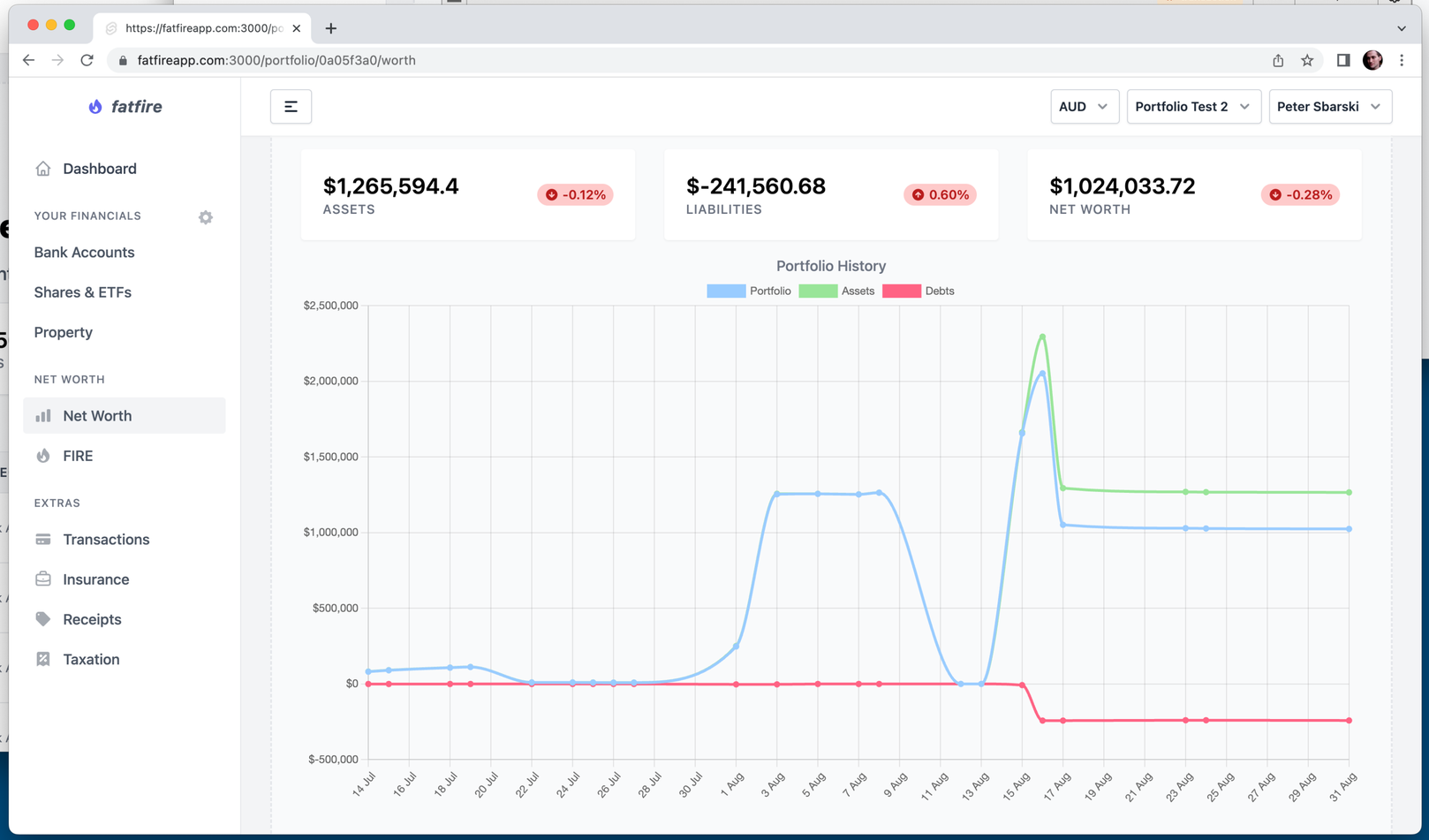
- Back to building on my own
- Serverless first approach makes technical decisions easier
- Have to be fast and reduce operational overhead to zero
- Could potentially need a lot of compute down the road
- Leverage as many AWS services as possible
- Solve local dev & debugging

fatfireapp.com

Wealth Operating System

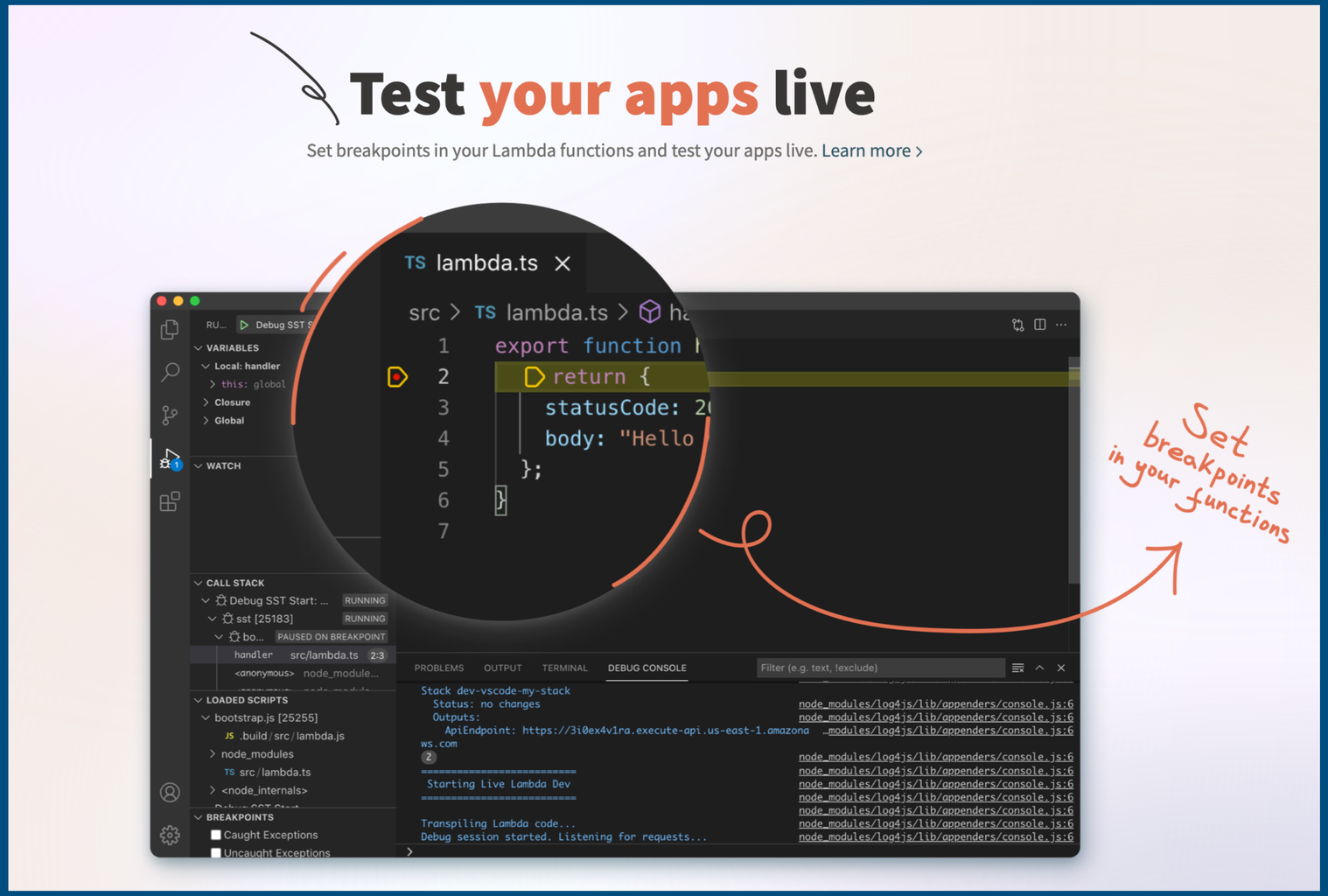






Serverless Stack (SST)

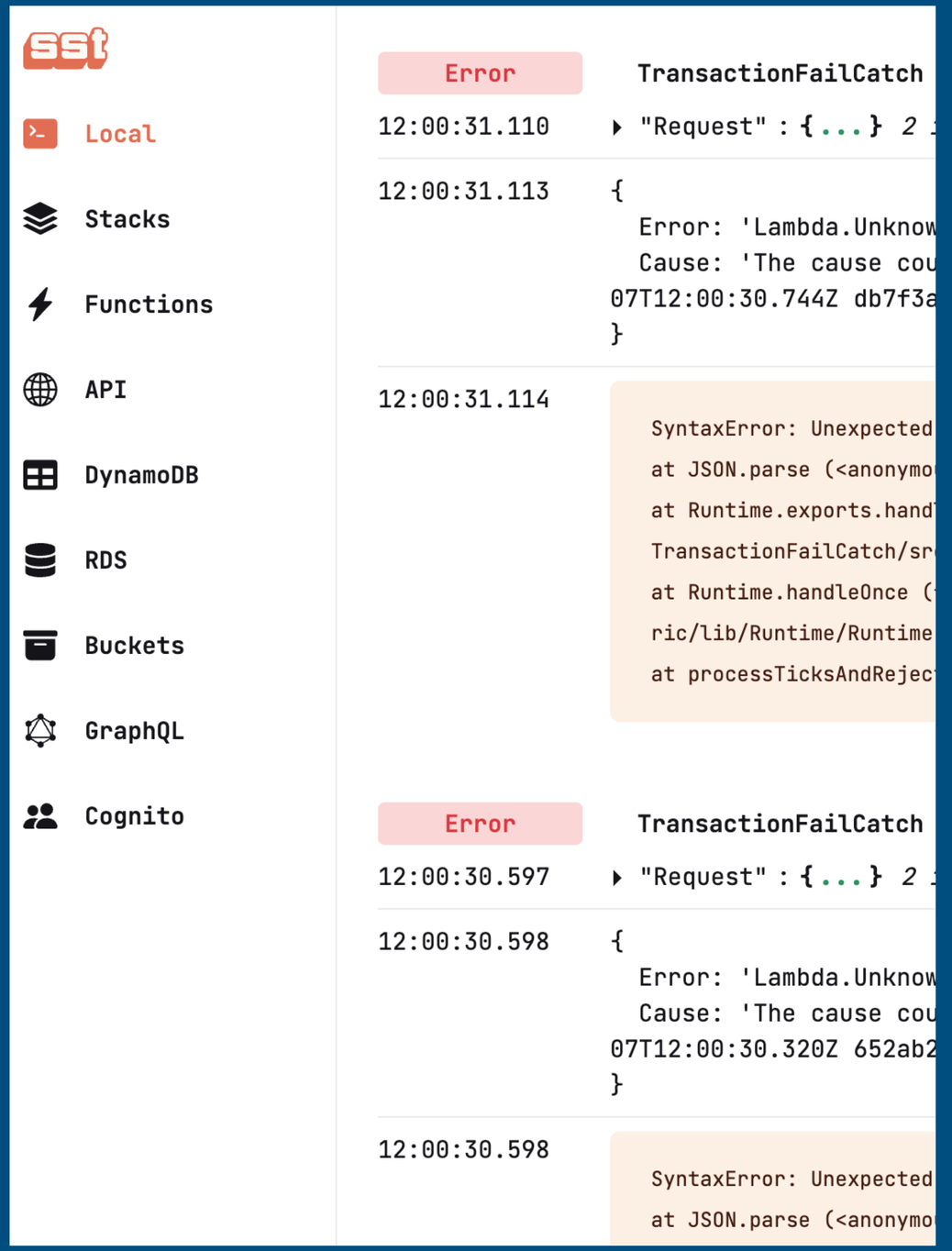
https://sst.dev



Serverless Stack

Quick overview

- Local dev & debugging
- Built on top of CDK
- Console for administration of your resources
- Higher constructs like Function and Config
- Auth (new!)



Serverless Stack - Auth

This is new

- Main components:
  - Auth - a construct that creates the necessary infrastructure
  - AuthHandler - a Lambda handler function that can handle authentication flows for various providers
  - Session - a library for issuing and validating authentication sessions in your Lambda function code.
- Source: https://docs.sst.dev/auth

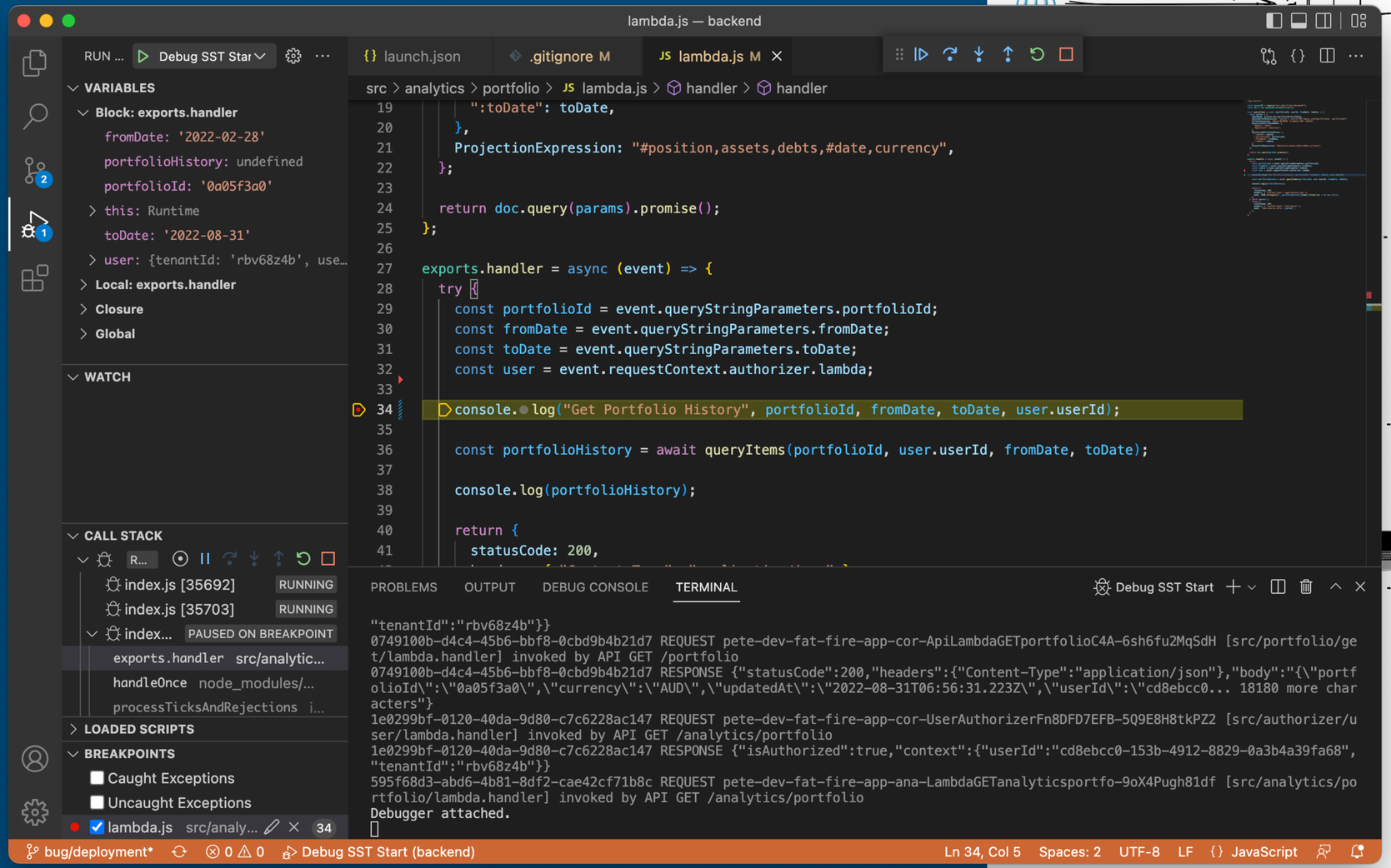
Authenticate through third-party providers

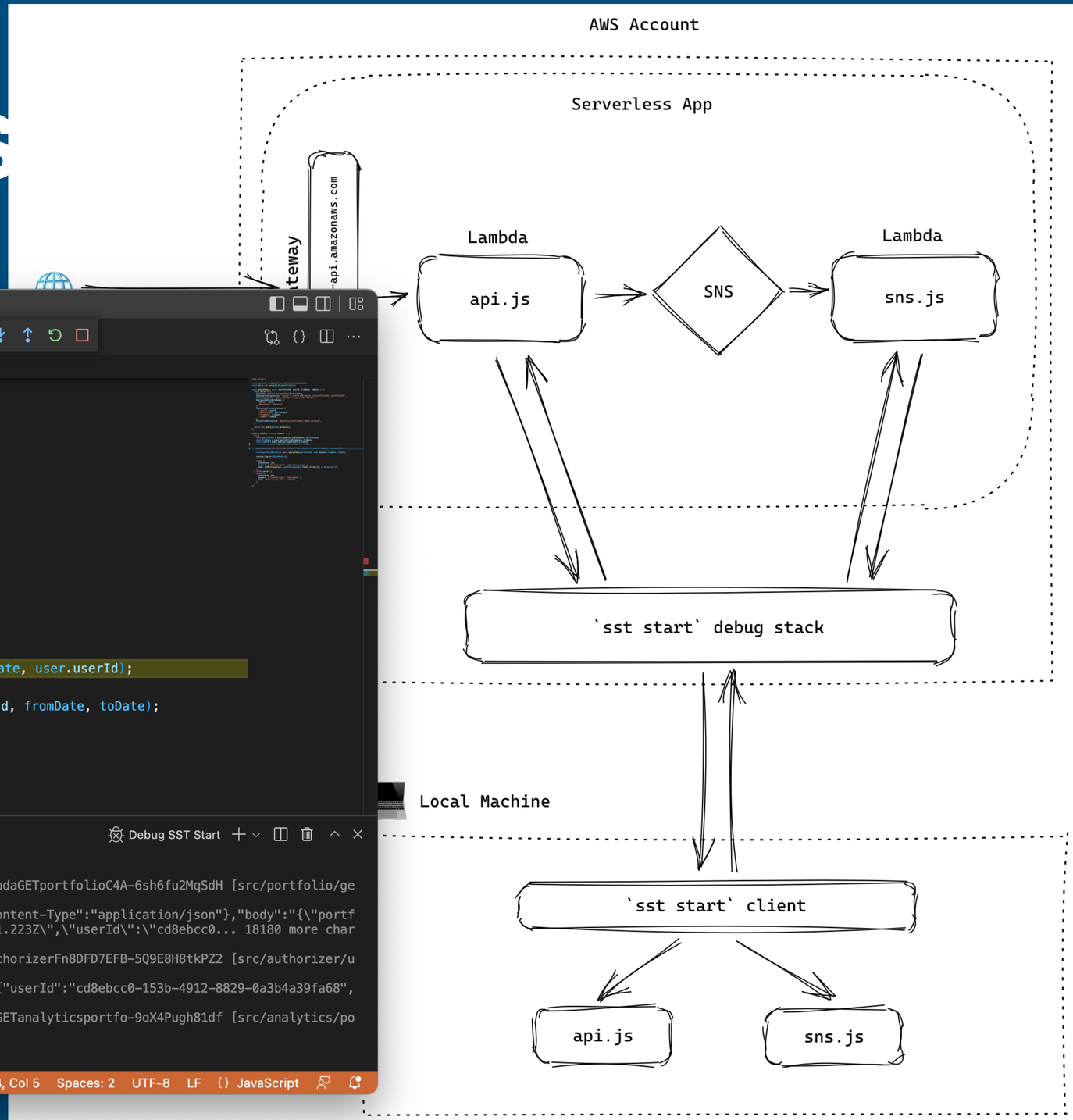
and/or send magic links

Don’t store or implement password functionality

Serverless Stack (SS

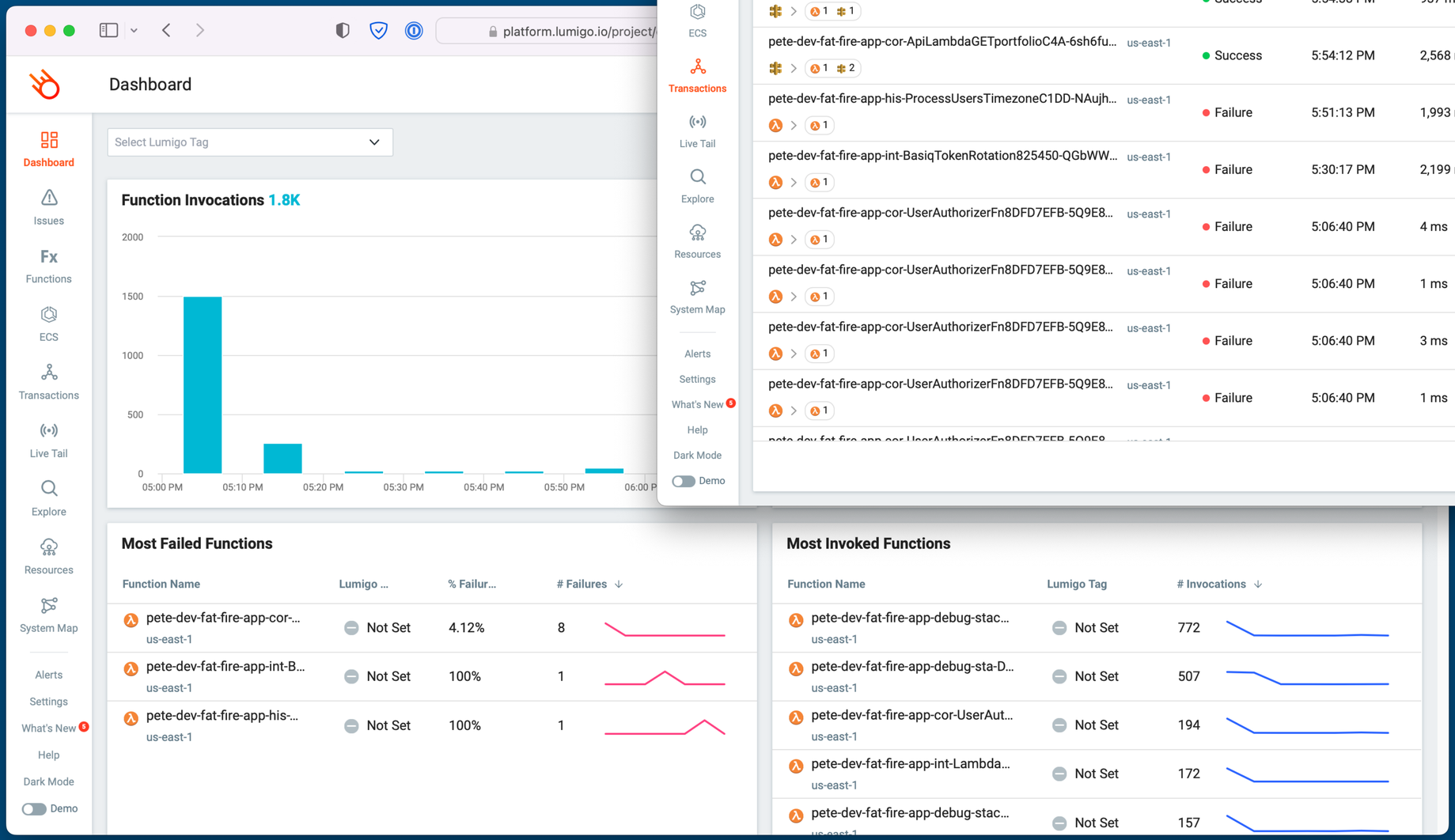
Local Dev. Absolutely Brilliant.

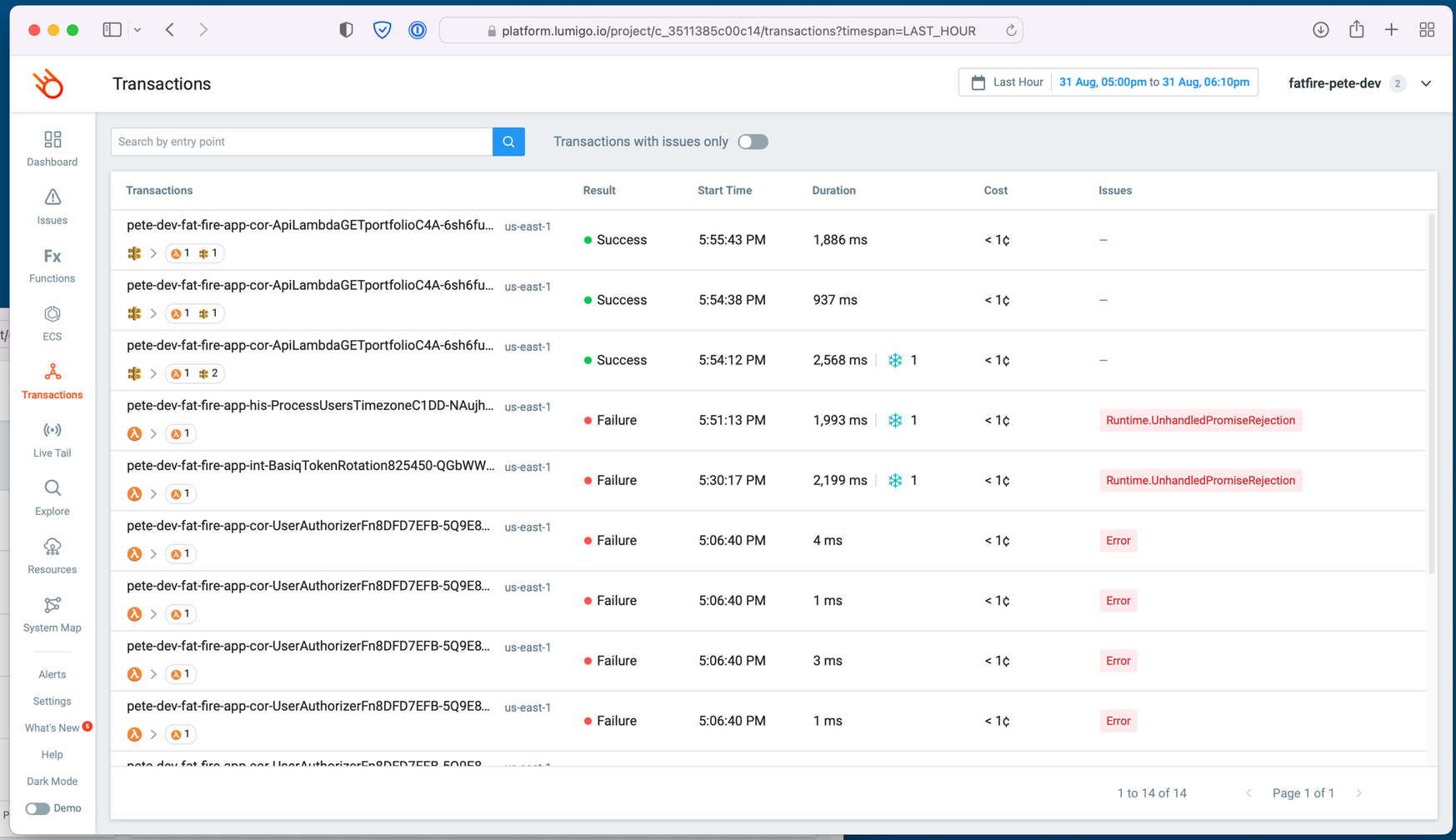




Observability

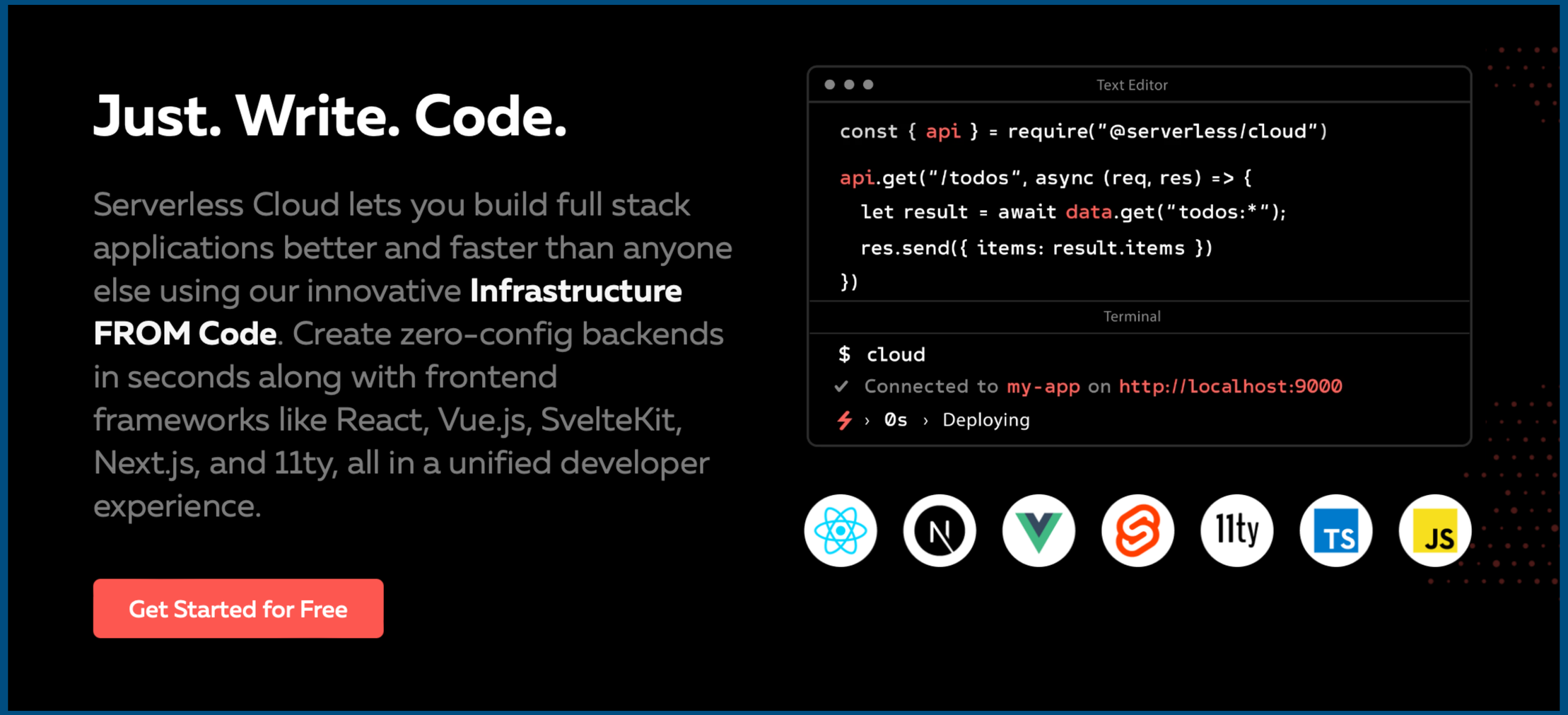
https://lumigo.io



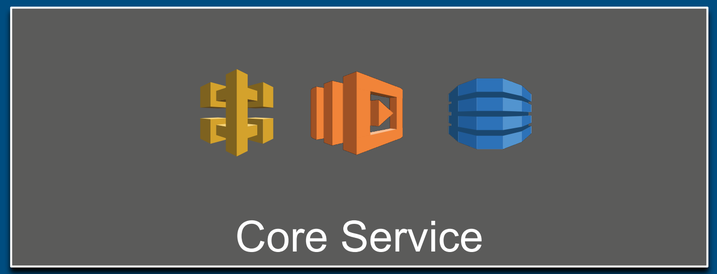


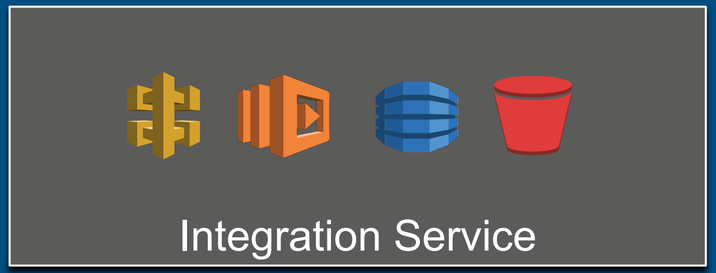
Serverless Cloud

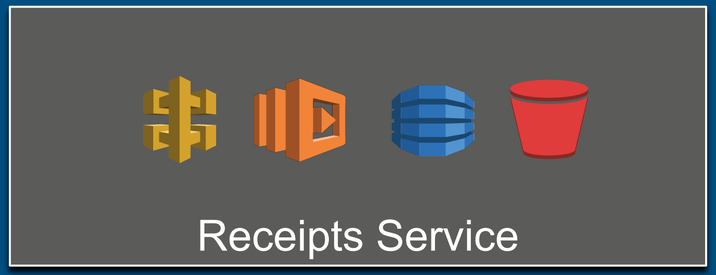
https://www.serverless.com/cloud



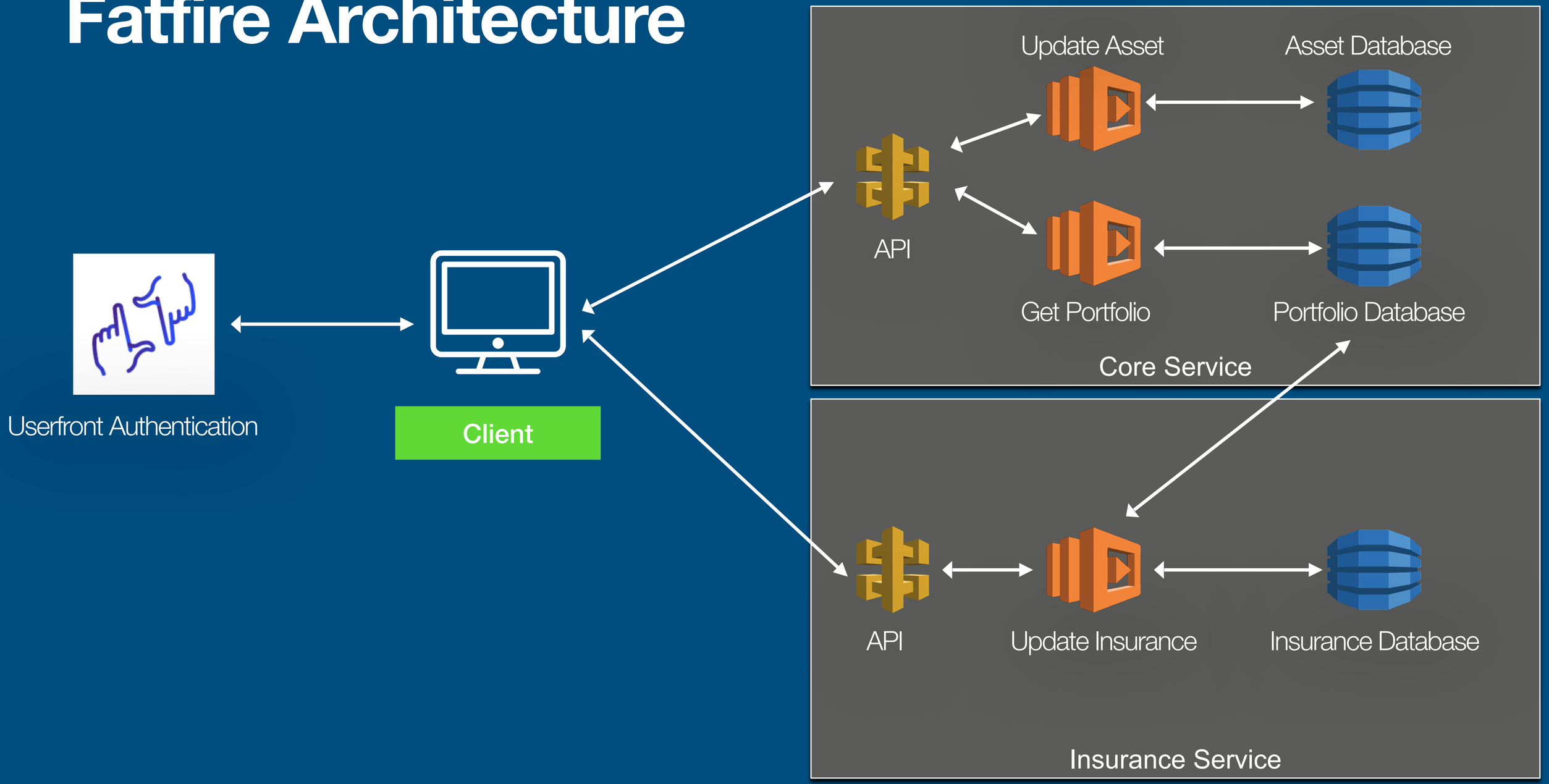
Fatfire Architecture

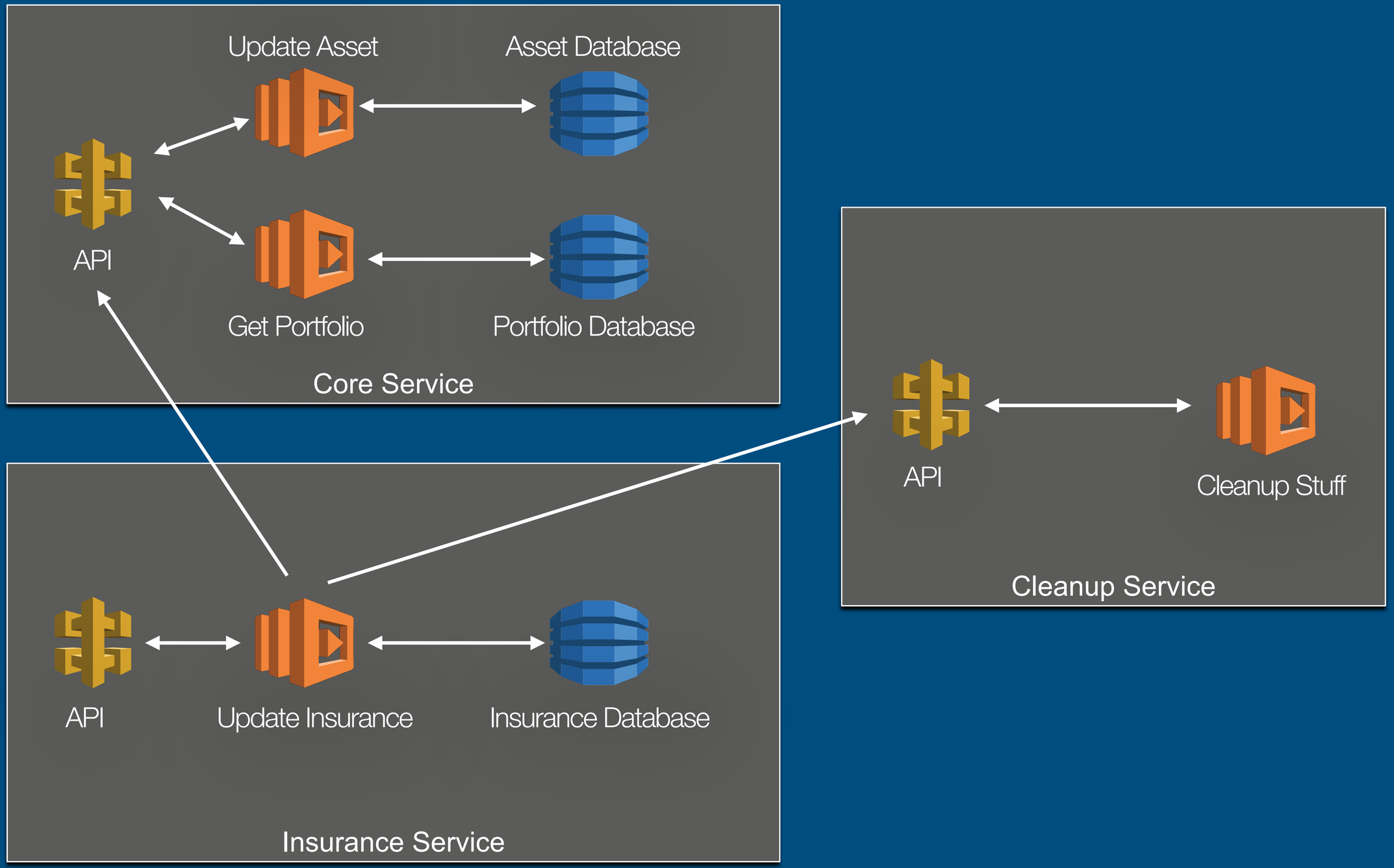


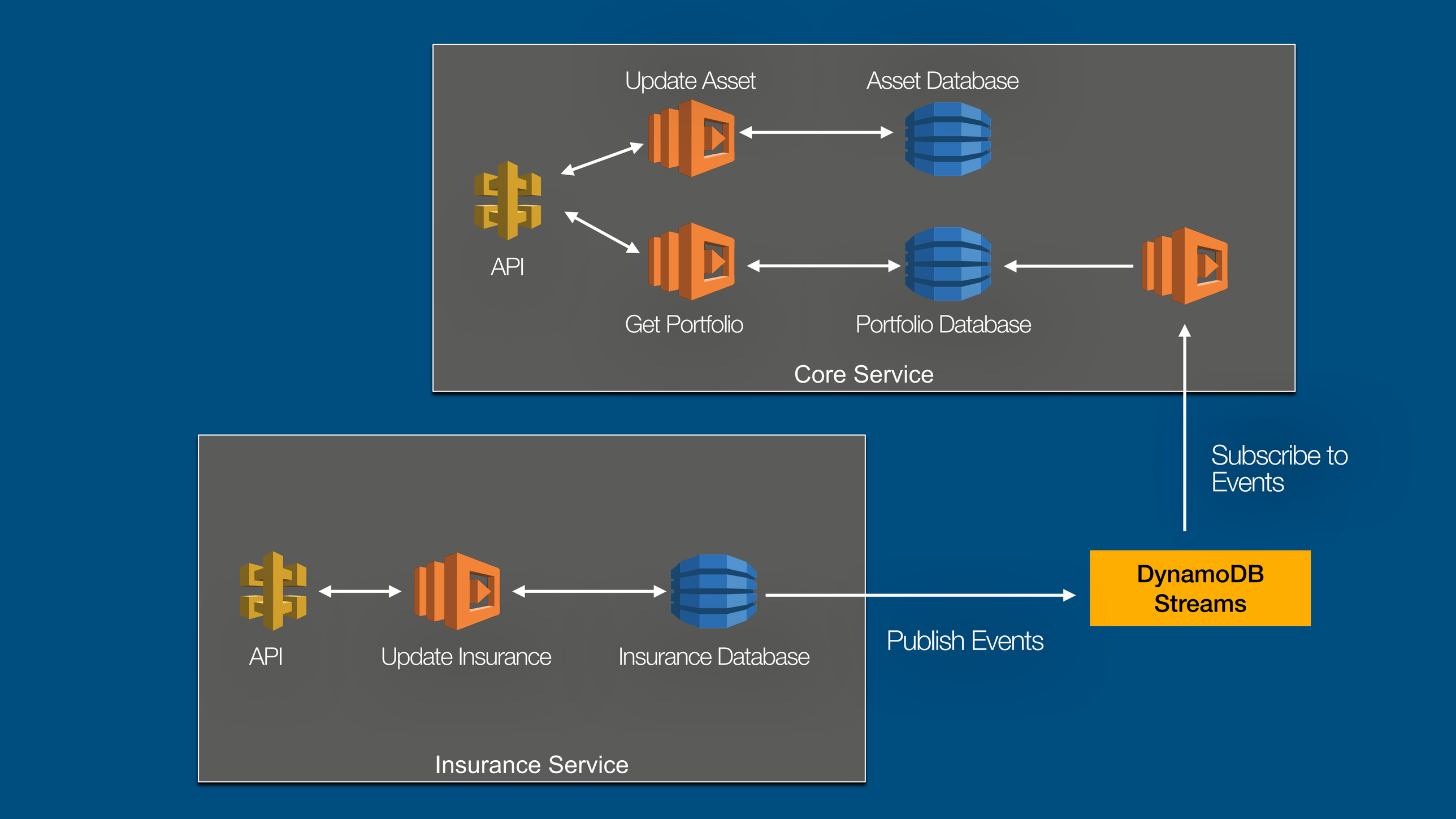




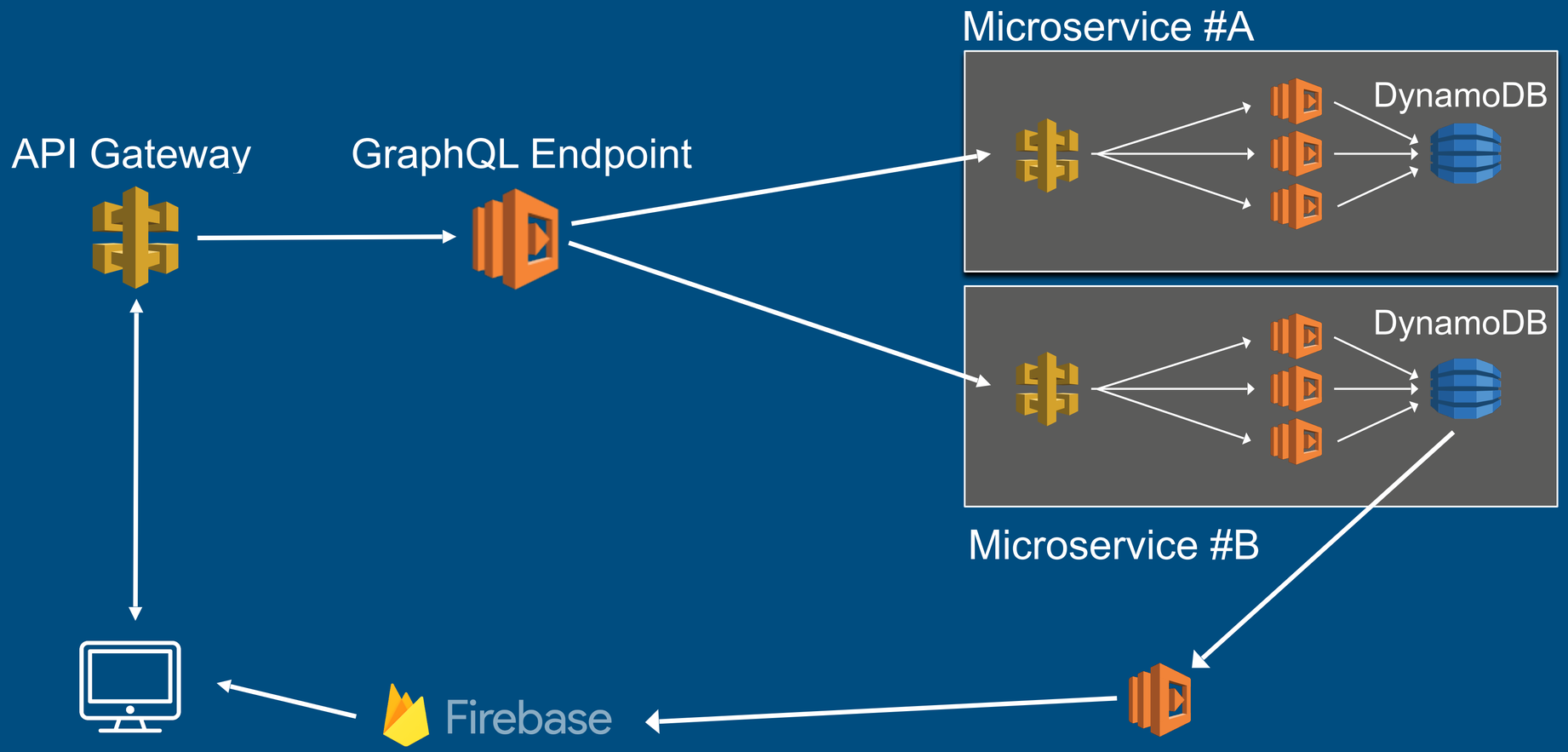
Fatfire Architecture

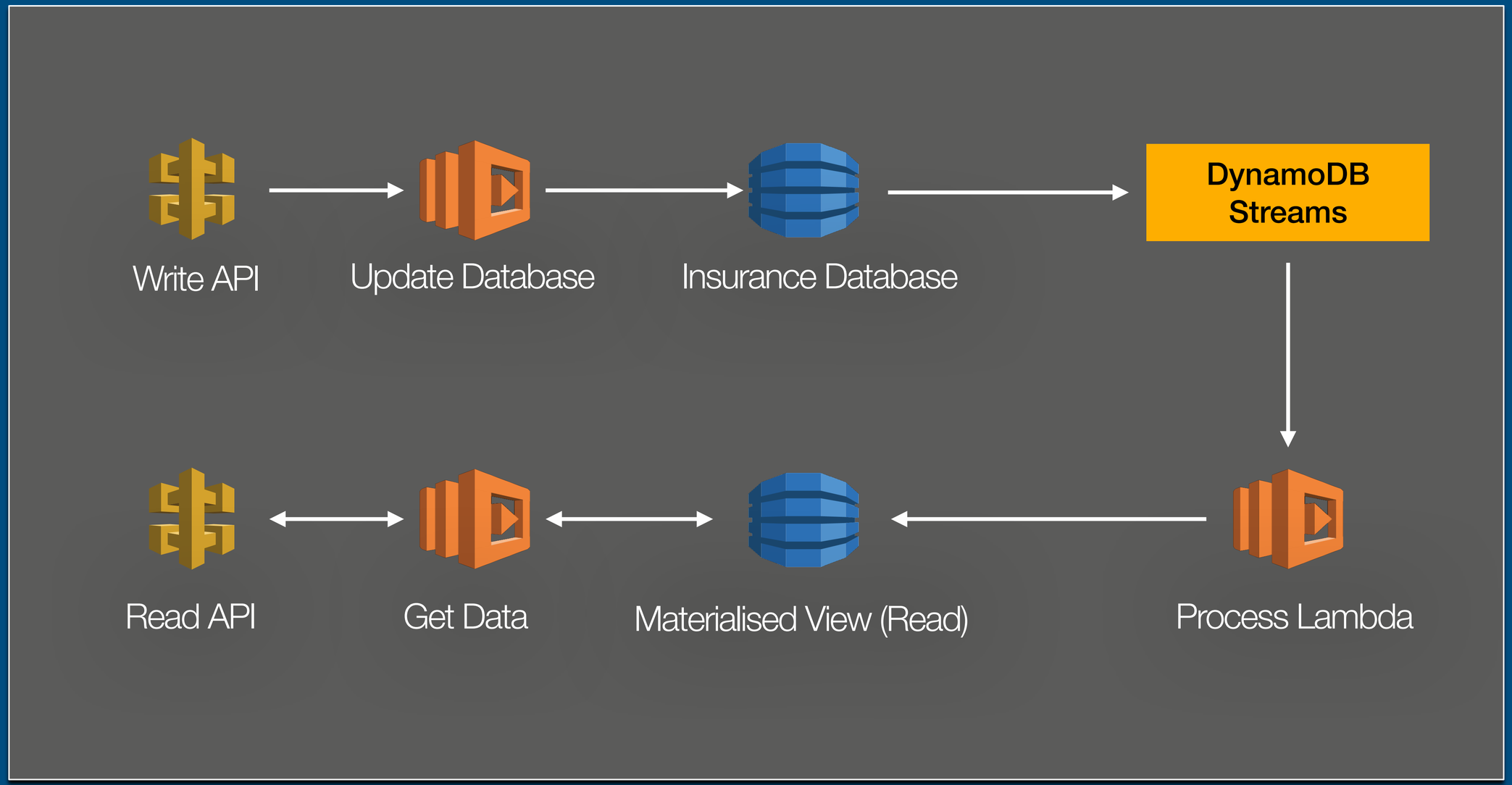


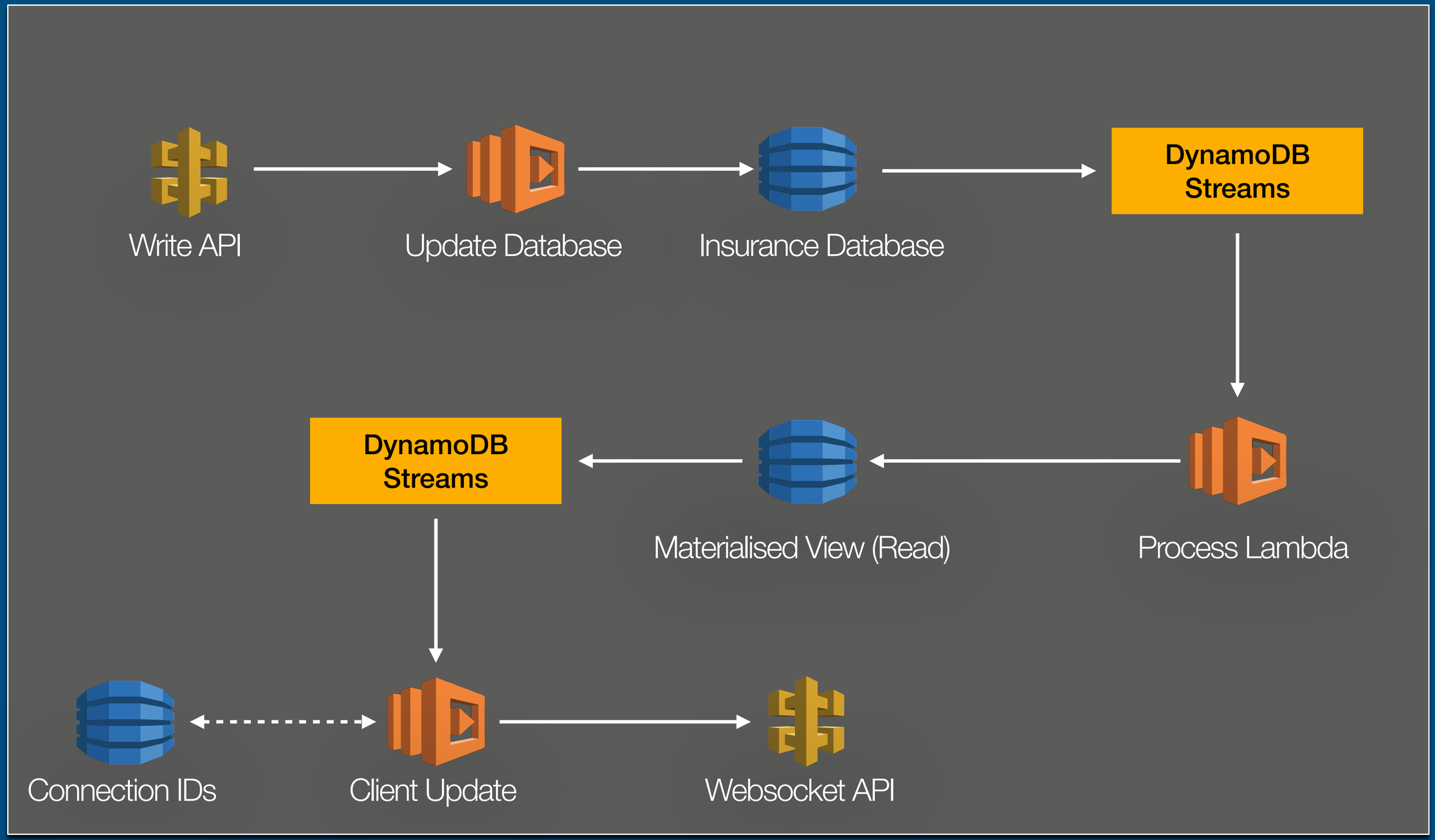

Update Asset
Asset Database
API
Get Portfolio
Portfolio Database
Core Service
API
Cleanup Stuff
Cleanup Service
API
Update Insurance
Insurance Database
Insurance Service

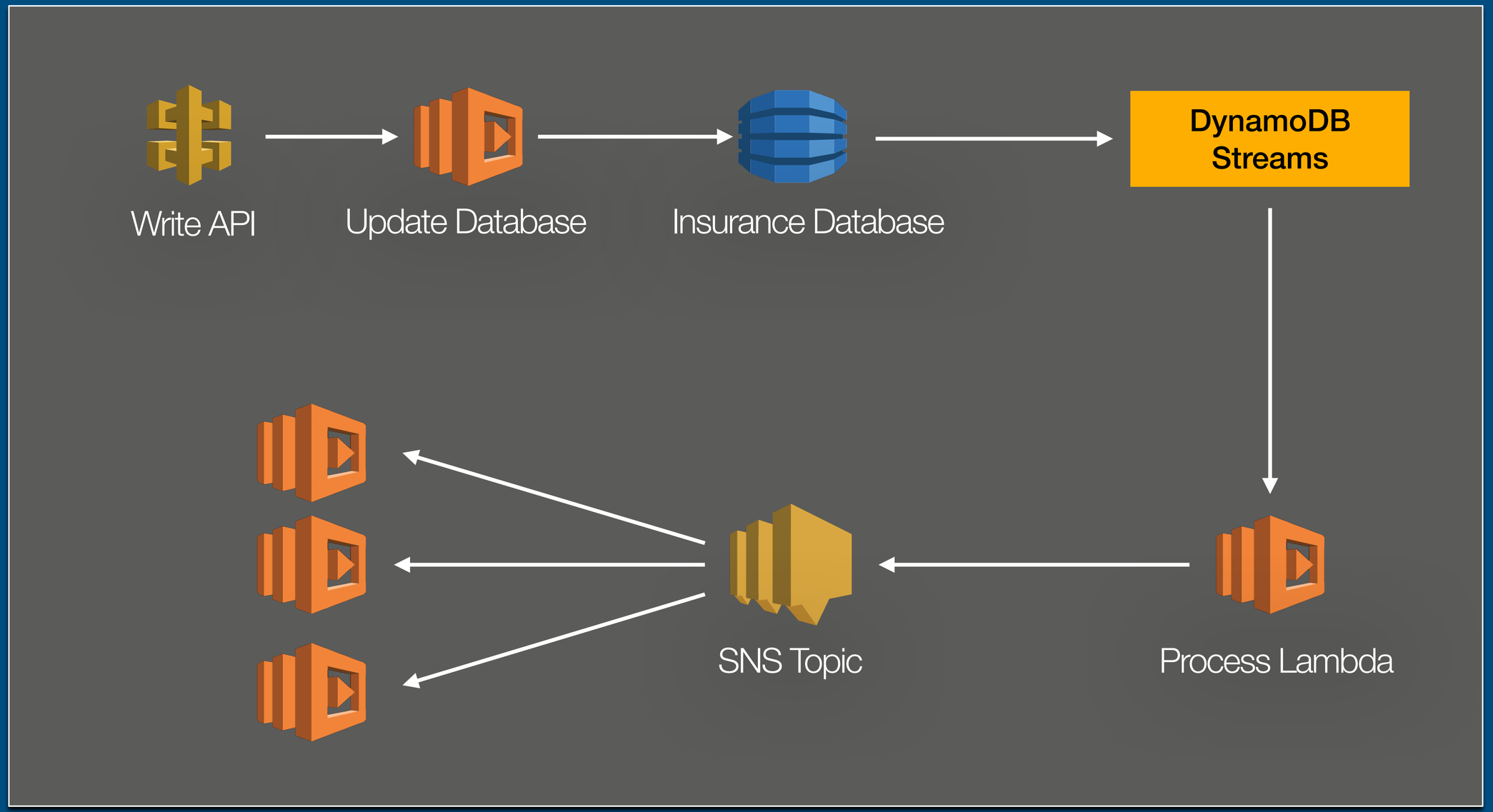

Update Asset
Asset Database
API
Get Portfolio
Portfolio Database
Core Service
Subscribe to Events
API
Update Insurance
Insurance Database
Insurance Service
Publish Events
DynamoDB Streams

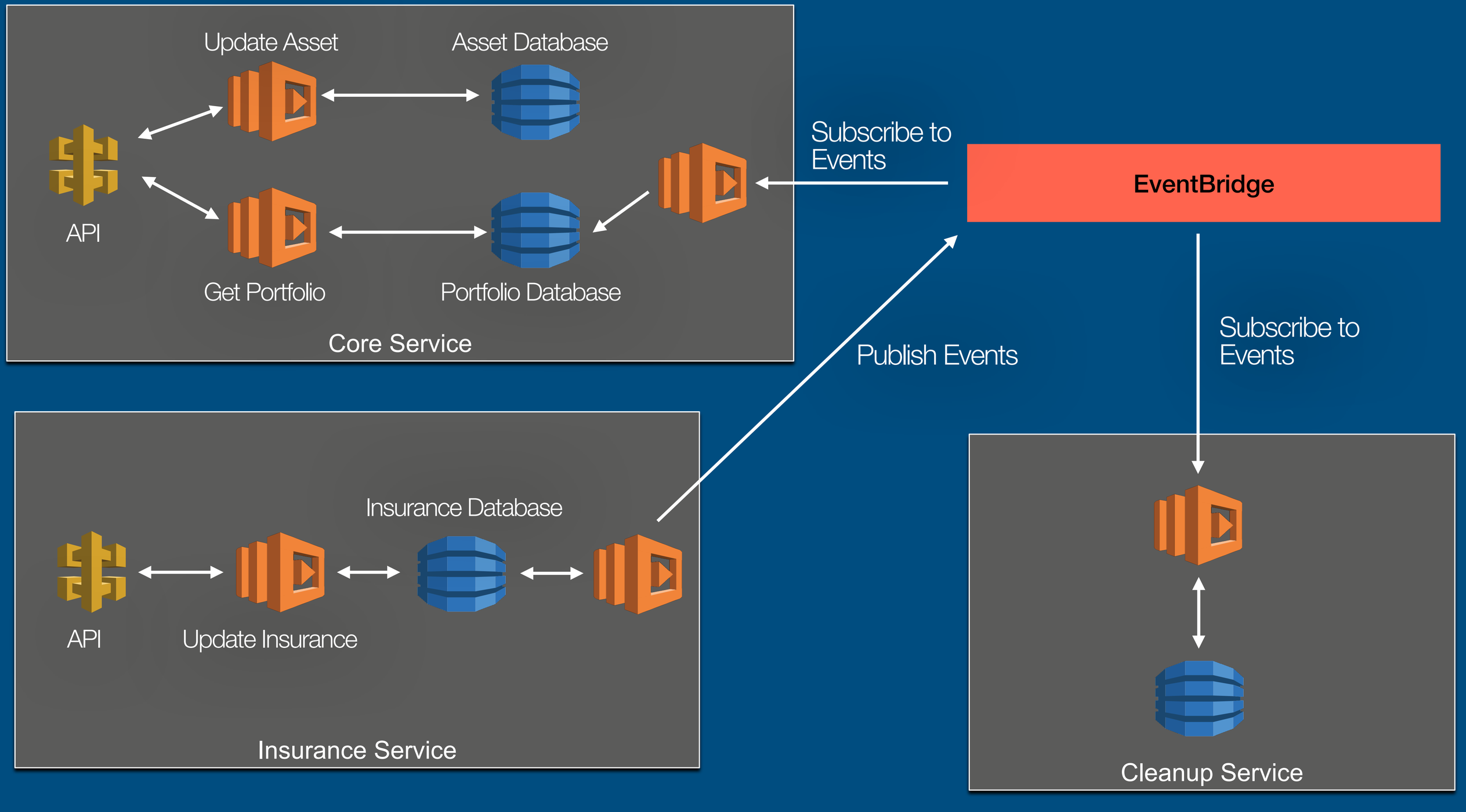
Remember this architecture?




Write API
Update Database
Insurance Database
DynamoDB Streams
Read API
Get Data
Materialised View (Read)
Process Lambda

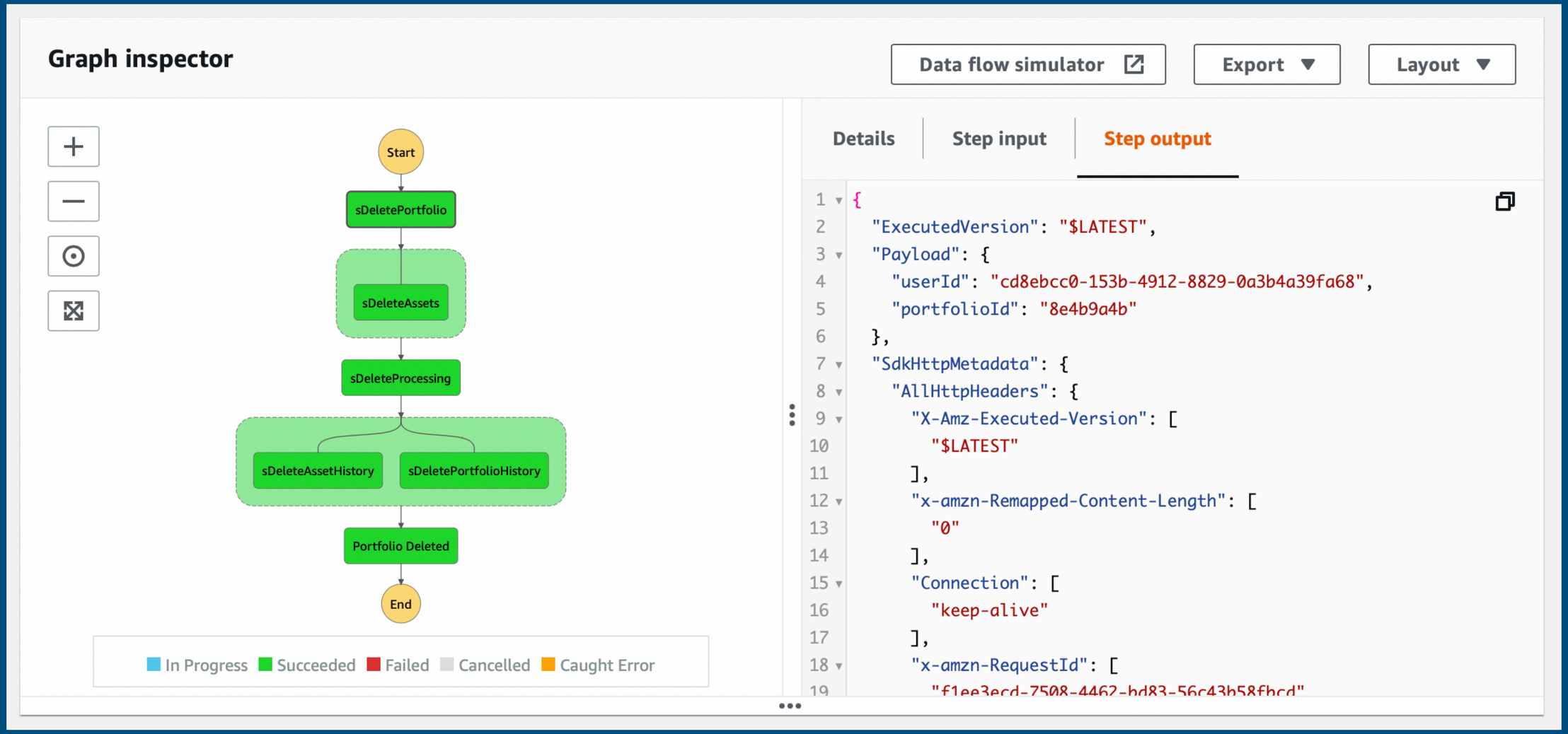

Write API
Update Database
Insurance Database
DynamoDB Streams
Process Lambda
Materialised View (Read)
DynamoDB Streams
Connection IDs
Client Update
Websocket API

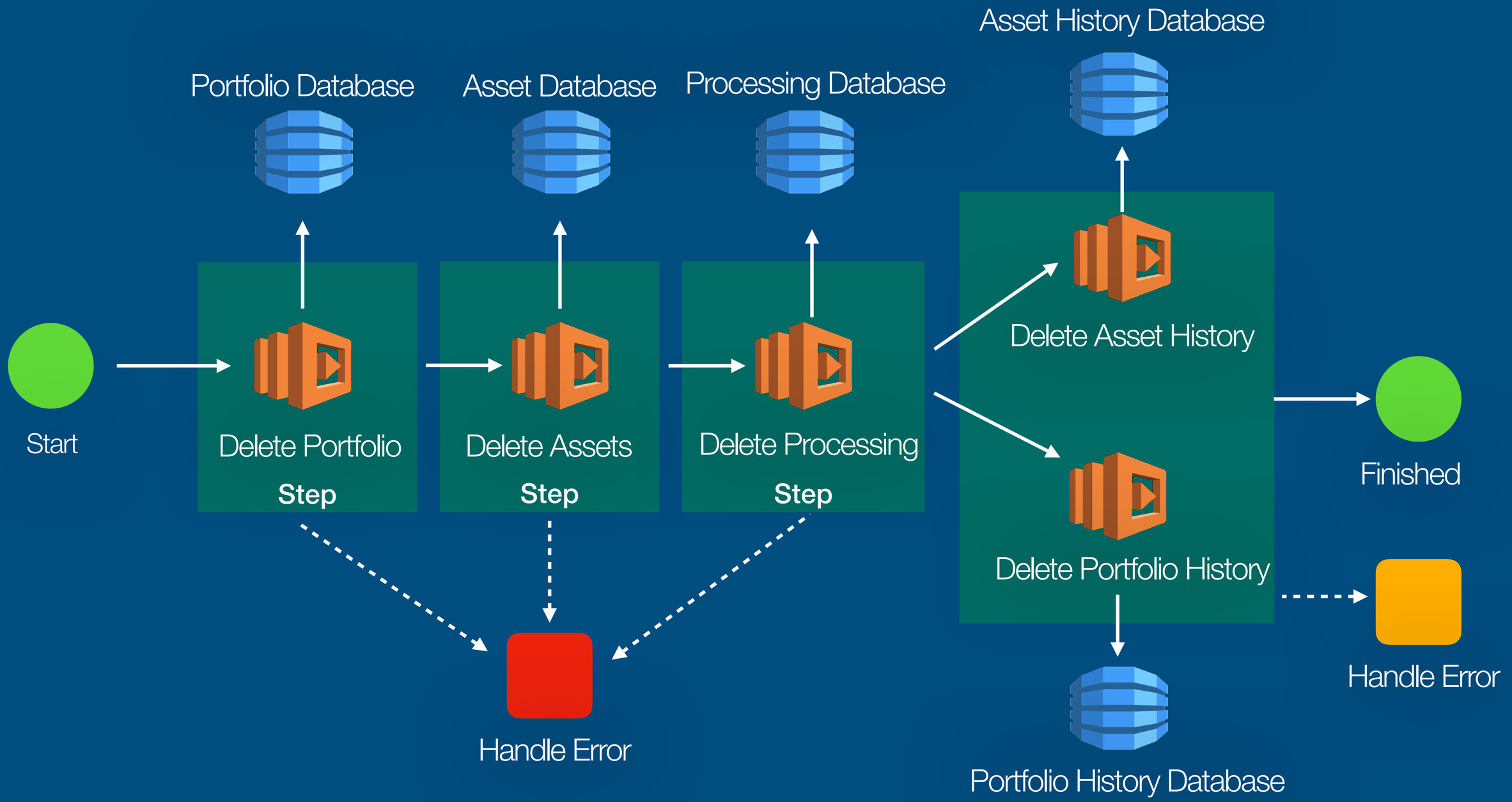

Write API
Update Database
Insurance Database
DynamoDB Streams
SNS Topic
Process Lambda


Update Asset
Asset Database
API
Get Portfolio
Portfolio Database
Core Service
Subscribe to Events
EventBridge
Publish Events
Subscribe to Events
Insurance Database
API
Update Insurance
Insurance Service
Cleanup Service

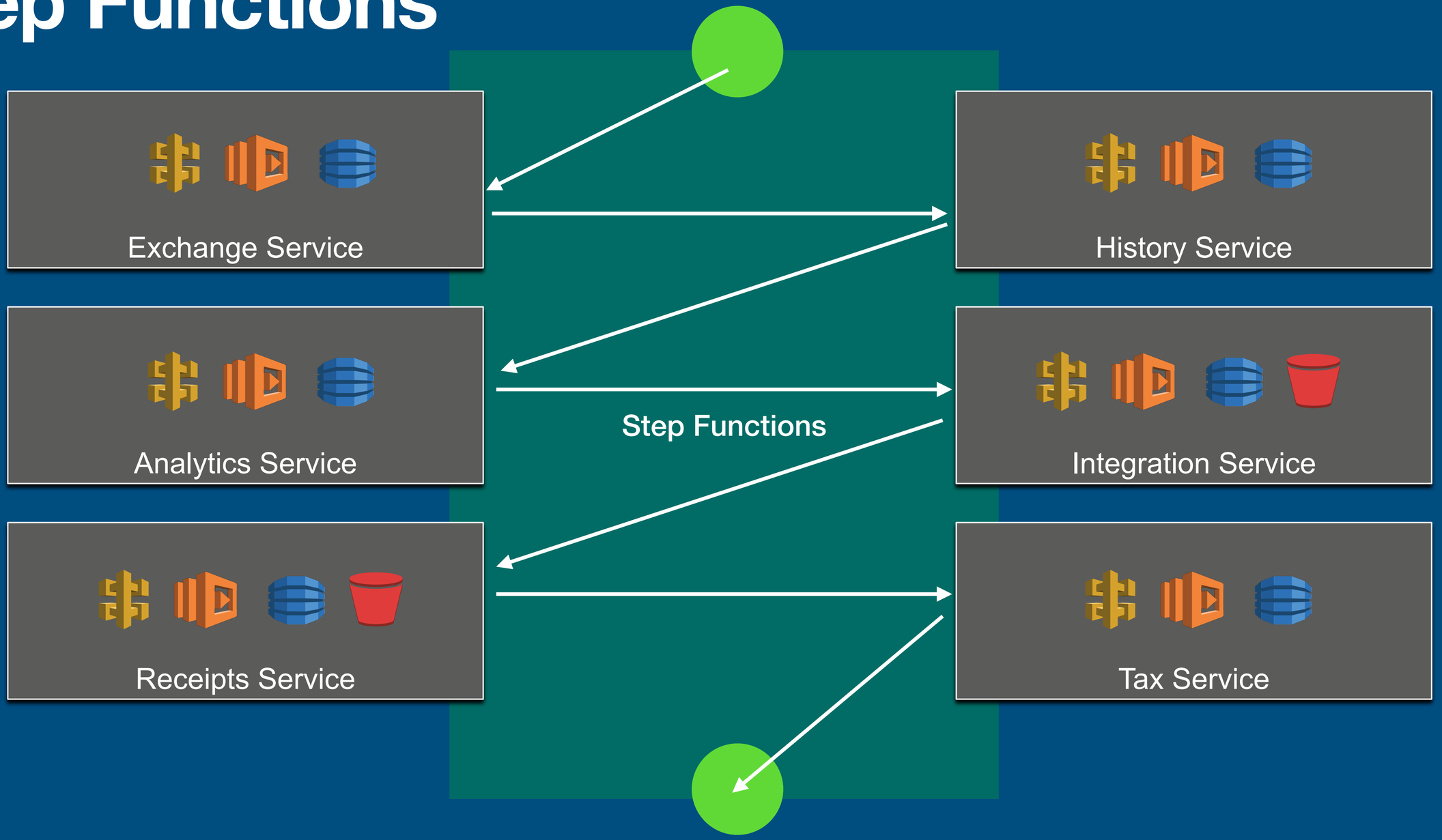
Step Functions

Coordinate Distributed Applications




Asset History Database
Portfolio Database
Asset Database
Processing Database
Delete Asset History
Start
Delete Portfolio
Step
Delete Assets
Step
Delete Processing
Step
Finished
Delete Portfolio History
Handle Error
Handle Error
Portfolio History Database

Step Functions



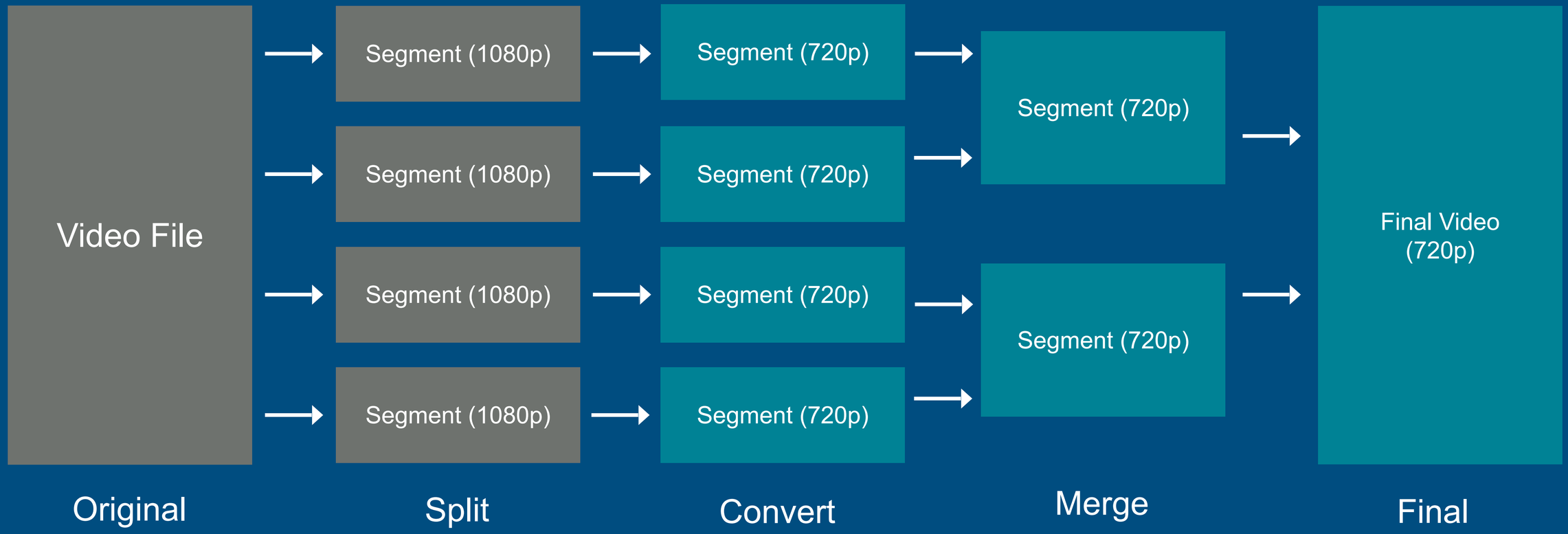
Parallel Architecture

Dealing with unexpected problems

- Take a complex problem and solve it with Lambda by applying techniques like MapReduce & Parallelisation
- Can you transcode (i.e. encode) a large video file with a Serverless-only approach?

Divide and conquer

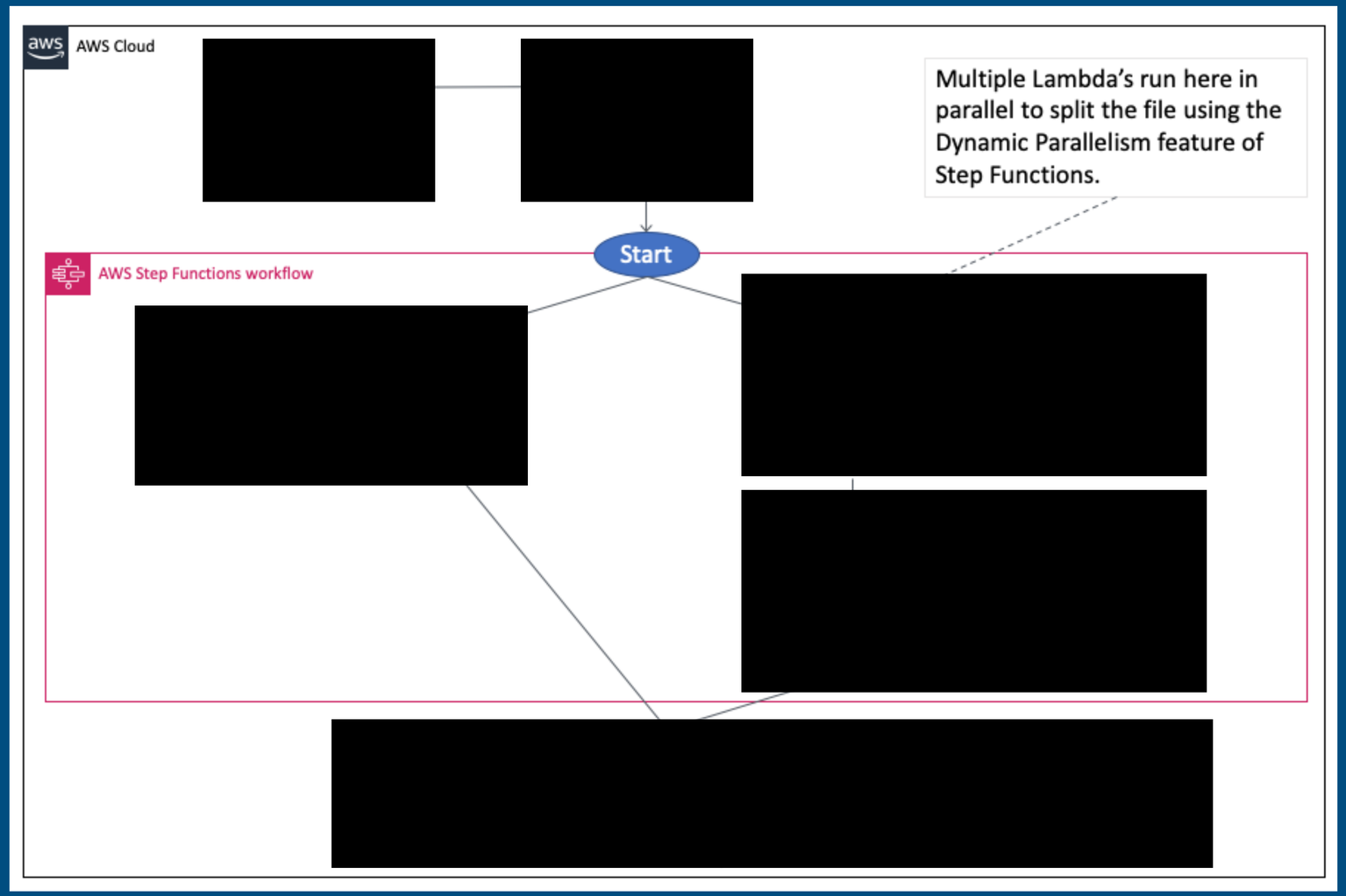
Using the Lambda supercomputer



Read more: https://bit.ly/3wJOdvQ

Parallel Computing with Lambda & Step Functions

Read more: https://bit.ly/3wJOdvQ



Serverless Video Transcoder

Parallel and conquer

	Serverless Lambda	Traditional EC2 (t2.large)	MacBook Pro 16GB 3.5GHz i7
34MB MP4 (00:43, 1920×1080)	11 seconds	32 seconds	18 seconds
77MB MP4 (6:49, 2048×1152)	26 seconds	144 seconds	78 seconds
100MB MP4 (59:56, 1280×720)	86 seconds	1073 seconds	592 seconds
350MB MP4 (07:45, 2560×1440)	35 seconds	432 seconds	224 seconds
420MB MKV (01:02, 3840 x 1606)	112 seconds	157 seconds	101 seconds
1GB MKV (57:57, 1280 x 718)	185 seconds	4320 seconds	2367 seconds

Read more: https://bit.ly/3wJOdvQ

Common Complaints

Why can’t things just be easy

- Hard to dev locally ✓
- Hard to debug ✓
- Hard to observe and monitor ✓
- Hard or impossible to do certain things (e.g. long-running tasks) ✓
- Lock-in is a problem ?

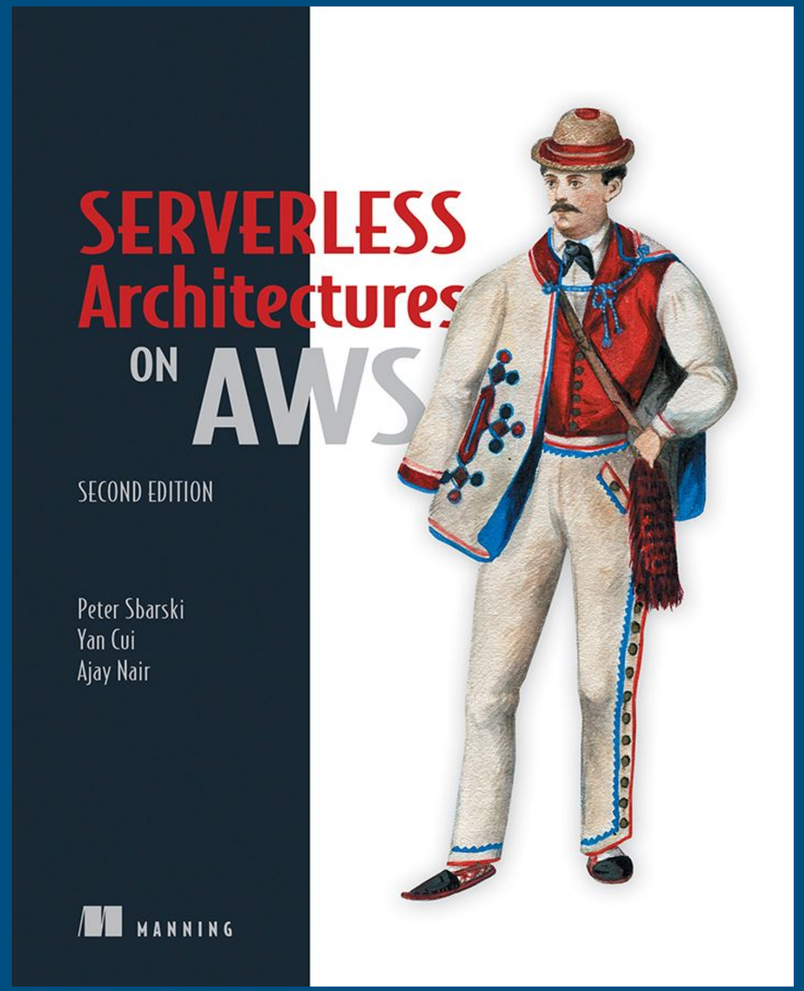
Modern Applications

Some lessons were learnt

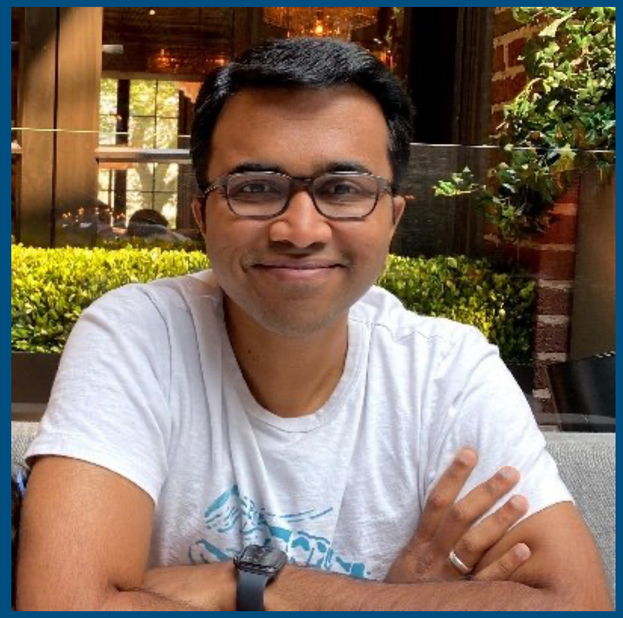
- Security/compliance first
- Use microservices
- Serverless where possible
- CI/CD
- Monitor, monitor, monitor!
- https://youtu.be/IPOvrK3S3gQ

- Serverless monoliths can be OK!
- Automation is a must
- Think through your testing strategy
- Experimentation and architectural changes are easier
- Serverless (& services) > containers

Thank you



Yan



Ajay

Book: https://mng.bz/z5mA

Fatfire: https://fatfireapp.com